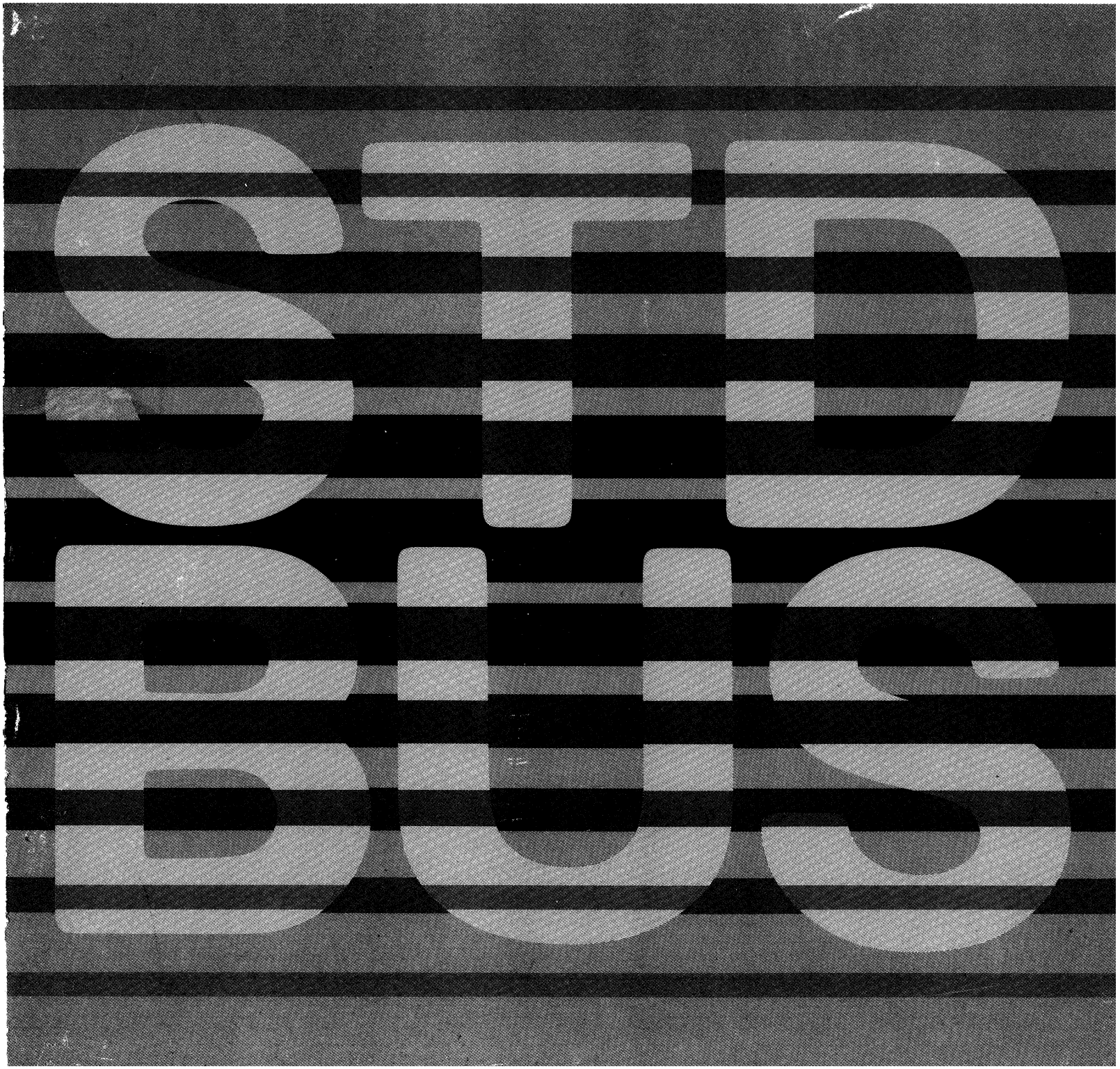
PRO-LOG®
CORPORATION

STD 7000

7862/8088 Processor Card

USER'S MANUAL



NOTICE

The information in this document is provided for reference only. Pro-Log does not assume any liability arising out of the application or use of the information or products described herein.

This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of Pro-Log, nor the rights of others.

Printed in U.S.A. Copyright © 1984 by Pro-Log Corporation, Monterey, CA 93940. All rights reserved. However, any part of this document **may be reproduced** with Pro-Log Corporation cited as the source.

STD 7000

7862

8088 Processor Card

USER'S MANUAL

FOREWORD

This manual explains how to use the Pro-Log 7862 8088 Processor Card. It is structured to reflect the answers to basic questions that you might ask about the 7862. We welcome your suggestions on how we can improve our instructions.

The 7862 is part of the Pro-Log Series 7000 STD BUS hardware. Our products are modular and designed and built with second-sourced parts that are industry standards. They provide industrial managers with the means of using their own people to control the design, production, and maintenance of products that use STD BUS hardware.

Pro-Log supports its products with thorough and complete documentation. Also, to provide maximum assistance to the user, we teach courses on how to design with and use microprocessors and the STD BUS products.

You may find the following Pro-Log documents useful in your work: the Microprocessor User's Guide and the Series 7000 STD BUS Technical Manual. If you would like a copy of these publications, please submit your request on your company letterhead.

CONTENTS

	Page
Foreword	ii
Figures	v
References and Bibliography	vii
Section 1. Purpose and Main Features	1-1
Purpose	1-1
Features	1-1
Functional Description	1-3
Section 2. Installation and Specifications	2-1
Introduction	2-1
Installation	2-1
Card Layout and Jumpers	2-2
Memory Configuration	2-6
Interrupt Vector Generation	2-7
Wait-State Insertion	2-8
Write Timing Selection	2-8
MEMEX and IOEXP Configuration	2-9
Timer Configuration	2-10
Serial Port Configuration	2-10
7862 STD BUS Interface	2-12
7862 User Interface	2-20
Specifications	2-22
Section 3. Operation and Programming	3-1
Introduction	3-1
Reset	3-1
Interrupts	3-1
I/O Devices	3-4
Serial Port	3-5
Initialization	3-5
Mode Byte	3-5
Command Byte	3-6
Status Byte	3-8
Programmable Interval Timer	3-9
System Memory Addressing	3-12
System I/O Addressing	3-14

Page

Section 4. Operating Software **4-1**

Introduction 4-1
Memory Addresses 4-1
I/O Port Addresses 4-1
Subroutine (INIT ASYNC COMM) 4-2
Subroutine (INIT SYNC COMM) 4-3
Subroutine (CHAR OUT) 4-4
Subroutine (CHAR IN) 4-5
Subroutine (SYNC BLOCK OUT) 4-6
Subroutine (SYNC BLOCK IN) 4-8
Subroutine (GEN CRC) 4-10
Subroutine (DELAY) 4-11
Subroutine (INIT INTR) 4-12
Subroutine (INTR) 4-13

Section 5. Maintenance **5-1**

Reference Drawings 5-1
Return for Repair/Warranty 5-6

Appendix A. The 8088 Microprocessor **A-1**

Introduction A-1
Architecture A-1
Operation A-7

Appendix B. The 8088 Instruction Set **B-1**

Introduction B-1
The 8088 Instruction Subset B-1
STD Instruction Mnemonics B-2

FIGURES

Figure		Page
1-1	7862 8088 Processor Card	1-2
1-2	Block Diagram	1-2
2-1	Installation	2-2
2-2	Jumper Configuration	2-2
2-3	Sources for Shorting Connector	2-3
2-4	Physical Layout and Jumper Options	2-4
2-5	Socket 0 Jumper Configuration	2-6
2-6	Socket 1 Jumper Configuration	2-6
2-7	Interrupt Vector Generator Jumper Options	2-7
2-8	I/O Wait-State Insertion Jumper Configuration	2-8
2-9	Write Timing Jumper Selection	2-8
2-10	MEMEX Configuration	2-9
2-11	IOEXP Configuration	2-9
2-12	Timer 2 Clock Source	2-10
2-13	Timer 2 Interrupts	2-10
2-14	Serial Port Configuration	2-10
2-15	Baud Rate Clock Selection	2-11
2-16	Serial Port Interrupt Configuration	2-12
2-17	STD Edge Connector Pin List	2-13
2-18	Bus Status Timing	2-14
2-19	Bus Read Timing	2-15
2-20	Bus Write Timing	2-16
2-21	Bus Interrupt Timing	2-17
2-22	Bus Wait Request Timing	2-18
2-23	Bus Exchange Timing	2-19
2-24	J1 Connector Pinout	2-20
2-25	Wire List for J1 to 25-Pin D Connector Cable	2-21
3-1	NMIRQ* Processing	3-2
3-2	INTRQ* Processing	3-2
3-3	I/O Port Address Map	3-4
3-4	USART Mode Byte Format	3-5
3-5	USART Command Byte Format	3-6
3-6	USART Status Byte Format	3-8
3-7	Baud Rate Selection	3-10
3-8	Sample System Memory Configuration	3-13
3-9	Sample SYSTEM I/O Port Configurations	3-15
4-1	(INIT ASYNC COMM) Flowchart	4-2
4-2	(INIT SYNC COMM) Flowchart	4-3
4-3	(CHAR OUT) Flowchart	4-4
4-4	(CHAR IN) Flowchart	4-5
4-5	(SYNC BLOCK OUT) Flowchart	4-7
4-6	(BY-SYNC) Protocol	4-7
4-7	(SYNC BLOCK IN) Flowchart	4-9
4-8	(GEN CRC) Flowchart	4-10
4-9	(DELAY) Flowchart	4-11
4-10	(INIT INTR) Flowchart	4-12
4-11	(INTR) Flowchart	4-13
4-12	Software Routines	4-14

Figures **Page**

5-1 Schematic 5-1
5-2 Assembly Drawing 5-3

A-1 Pipeline Structure A-1
A-2 8088 Register Set A-2
A-3 General Registers A-3
A-4 Implicit Use of General Registers A-4
A-5 Control Registers A-6
A-6 Segment Registers A-6
A-7 RESET Initialization States A-7
A-8 Segment Mapping A-8
A-9 Segment Starting Address A-9
A-10 8088 Address Components A-10
A-11 8088 Interrupting Sources A-10
A-12 Interrupt Vector Table A-12
A-13 8086/8088 System Data Flow A-13

B-1 8085 Register Subset B-1
B-2 Examples of Instruction Mnemonic Structure B-2
B-3 8088 Programming Aid B-40
B-4 8088 Structured Mnemonic Cross Reference B-43

REFERENCES/BIBLIOGRAPHY

STD BUS Specification and Practice, (Pro-Log Document #106989).

Series 7000 STD BUS Technical Manual, (Pro-Log Document #107855).

Microprocessor User's Guide, (Pro-Log Document #101145).

iAPX 88 Book, with an Introduction to the 188. Reston Publishing Company, Inc., Reston, Virginia. #R3016-4.

iAPX 86/88, 186/188 User's Manual — Programmer's Reference. Reston Publishing Company, Inc., Reston, Virginia. #R3035-4.

PURPOSE

The 7862 processor card combines the 16-bit capabilities of the 8088 CPU with two memory sockets for up to 72K bytes of onboard memory, an RS-232-C synchronous or asynchronous serial communications port, and a timer for software delays and timed interrupts. The buffered and fully expandable 8088 microprocessor operates at 4.9152 MHz and directly addresses up to 1 megabyte of memory and up to 64K bytes of I/O ports.

The two memory sockets conform to the 28-pin JEDEC standard and are mapped for the basic requirements of the 8088 microprocessor. One socket accepts a 2K or 8K byte-wide RAM, or a 4K byte-wide ROM and is mapped to support the 8088 interrupt structure. The second socket accepts 8K, 16K, 32K, or 64K byte-wide ROMs and is mapped to include the 8088 vector address following reset. Both memory sockets can be individually jumper disabled to allow unused address space to be mapped elsewhere in the system.

The 7862 drives the memory expansion (MEMEX) and I/O expansion (IOEXP) signals. These signals enable downward compatibility to STD memory cards that do not decode all 20 address lines and to STD I/O cards that do not decode 16 address lines. An onboard interrupt vector generator can be jumper selected to generate one of four vectors. The interrupt generator can also be jumper disabled if another interrupt vector generator is available in the system.

FEATURES

- 8088 CPU
- STD 8088 compatible
- 4.9152 MHz MIN mode operation
- Up to 64K bytes of onboard ROM
- Up to 8K bytes of onboard RAM
- Direct addressing of 1 megabyte of memory
- Direct addressing of 64K I/O ports
- Asynchronous or binary synchronous RS-232-C serial port with software selectable baud rate
- Two timers for software delays and timed interrupts
- Power-on and pushbutton reset
- MEMEX and IOEXP support

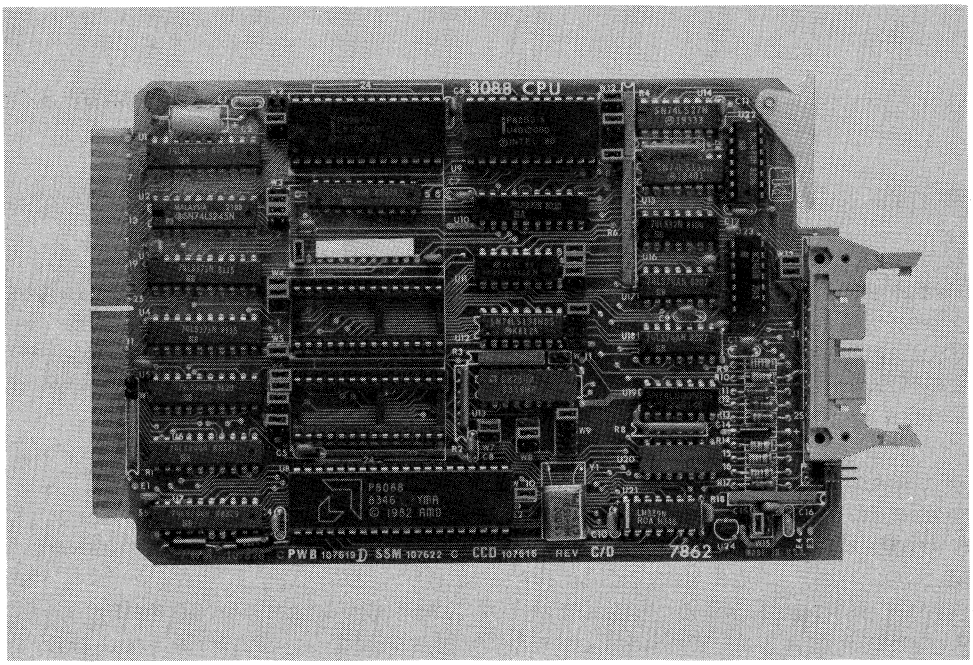


Figure 1-1. 7862 8088 Processor Card

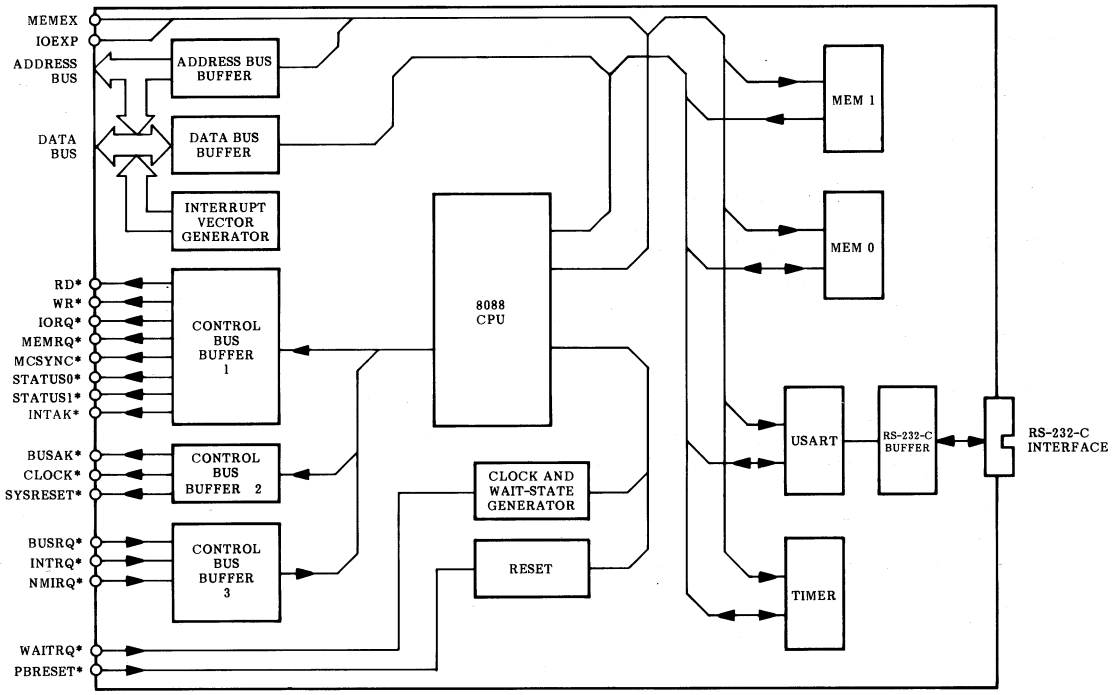


Figure 1-2. Block Diagram

FUNCTIONAL DESCRIPTION

This description defines each block in the diagram on page 1-2. The description and the diagram familiarize the user with terms used throughout this manual.

Address Bus Buffer — Provides buffering and 3-state control of the address bus. Address lines A0-A15 are latched. Address lines A16-A19 are time multiplexed onto data bus lines D0-D3. The rising edge of MCSYNC* is provided for latching A16-A19.

Data Bus Buffer — Provides buffering, direction, and 3-state control of the data bus (D0-D7). The lower four data lines (D0-D3) are time multiplexed with the upper four address lines (A16-A19).

Interrupt Vector Generator — Provides a jumper selectable interrupt vector for use by the CPU during interrupt servicing. Jumpers permit selection of one of four vector types.

Control Bus Buffer 1 — Provides buffering and 3-state control of memory, I/O, and peripheral control signals.

Control Bus Buffer 2 — Provides buffering for bus control signals that do not require 3-state capabilities. These signals are BUSAK*, CLOCK*, and SYSRESET*.

Control Bus Buffer 3 — Provides buffering for bus control lines that are inputs to the card. These signals are INTRQ*, NMIRQ*, and BUSRQ*.

Reset — Provides a signal to place the card in a reset state during power-up and pushbutton reset (PBRESET* active).

Clock and Wait-State Generator — Generates a 4.9152 MHz CPU clock and permits jumper selectable wait-state insertion. Through jumper configuration, a single wait state can be inserted for access to I/O devices off-board. WAITRQ* overrides this function.

MEM 1 — A 28-pin socket for user-supplied memory. It can be configured for an 8K x 8, 16K x 8, 32K x 8, or a 64K x 8 EPROM. The EPROM's highest address is at FFFFF.

MEM 0 — A 28-pin socket for user-supplied memory. It can be configured for either a 2K x 8 or an 8K x 8 RAM or a 4K x 8 EPROM. The memory's lowest address is at 00000.

USART — Provides for asynchronous or binary synchronous serial communications.

RS-232-C Buffer — Translates the TTL signals from the USART into RS-232-C levels and vice versa.

Timer — The timer circuit has 3 independent 16-bit timers. The first timer generates the baud rate clock for the USART. The second is connected to the 8088's TEST* pin and can be connected to the input clock to the third timer. The third timer can be used to generate periodic interrupts.

INTRODUCTION

This section describes the procedure for installing the 7862 card into an STD BUS system. Options and specifications are included that must be considered before operation.

INSTALLATION

CAUTION

To prevent possible damage to your STD BUS system, make sure power is **off** before inserting a card into the card rack or before removing a card from the card rack.

CAUTION

To guarantee the proper operation of the 7862 8088 Processor Card, care must be taken to ensure that the LOGIC GND and AUX GND are connected at the user's power supply.

Pro-Log power supplies are shipped with both GNDs bused together and therefore meet these requirements. If a source other than Pro-Log is used for power supplies, then care must be taken to ensure that these requirements are met.

The 7862 operates as part of an STD BUS system. The 7862 can be inserted into any slot in the STD BUS card rack. The ejector tab must be at the top as illustrated in figure 2-1. If access to the card is needed for testing or servicing, a card extender such as the Pro-Log 7901 may be used.

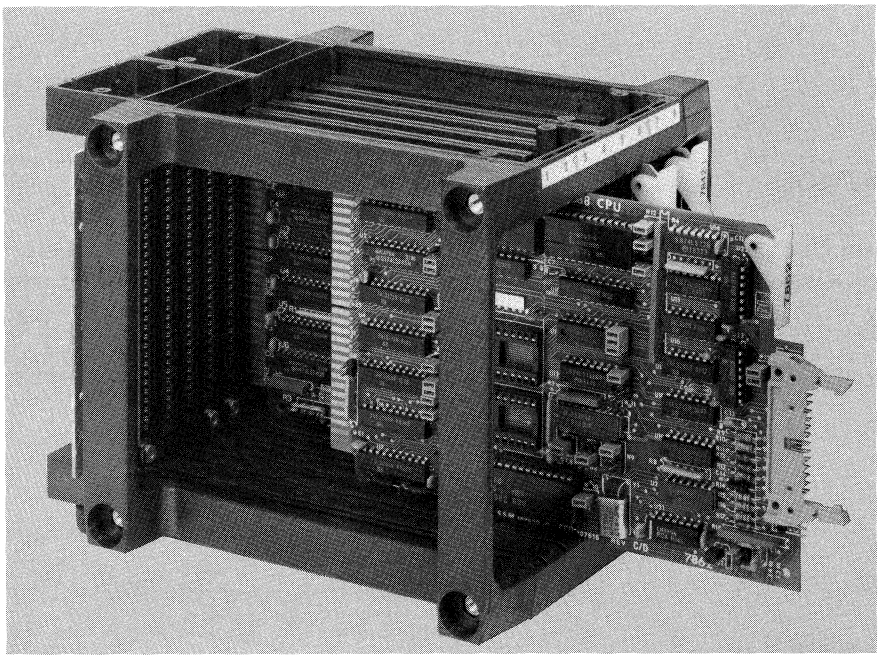


Figure 2-1. Installation (7862)

CARD LAYOUT AND JUMPERS

The 7862 has onboard jumpers to configure the card to meet system requirements. The jumper configurations consist of unshrouded headers and shorting connectors, as shown in figure 2-2. The shorting connectors can be removed and installed without the use of soldering equipment. Figure 2-3 lists sources for the shorting connectors.

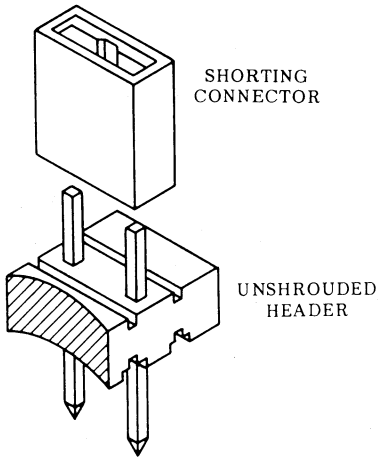


Figure 2-2. Jumper Configuration (7862)

SOURCE	PART NUMBER
Pro-Log	902520
Elco	008261024200870
Amp	531220-3

Figure 2-3. Sources for Shorting Connector (7862)

Figure 2-4 on the following pages provides a reference for the location and a brief description of each jumper on the 7862. The following discussion divides the jumper options into five functional groups: Memory Configuration, Interrupt Vector Generation, Wait-State Insertion, Write Timing Selection, MEMEX and IOEXP Configuration, Timer Configuration, and Serial Port Configuration.

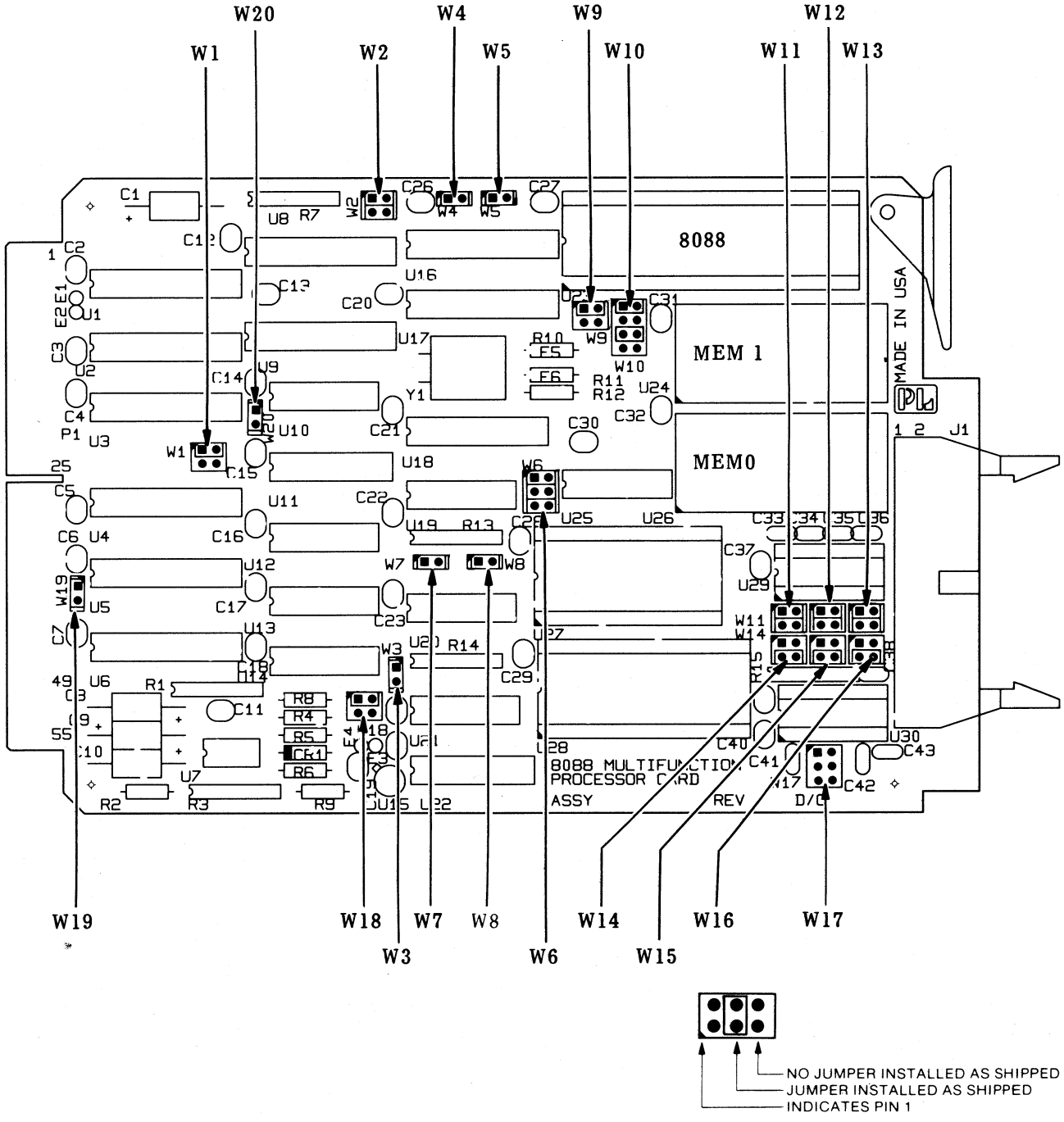


Figure 2-4. Physical Layout and Jumper Options (7862)

JUMPER GROUP	JUMPER INSTALLED	JUMPER REMOVED	DESCRIPTION	AS SHIPPED	SEE PAGE
W1	— 1-2 3-4	— 3-4 1-2	ADVANCED/NORMAL WRITE SELECT o Normal Write o Advanced Write	*	2-8
W2	1-2,3-4 1-2 3-4 —	— 3-4 1-2 1-2,3-4	INTERRUPT VECTOR SELECT o Vector 05H o Vector 25H o Vector 45H o Vector 65H	*	2-7
W3	1-2 —	— 1-2	I/O WAIT STATE INSERTION o Enable o Disable	*	2-8
W4	1-2 —	— 1-2	MEM 0 ENABLE o Enable o Disable	*	2-6
W5	1-2 —	— 1-2	MEM 1 ENABLE o Enable o Disable	*	2-7
W6	1-2,3-4,5-6 1-2,3-4 1-2 —	— 5-6 3-4,5-6 1-2,3-4,5-6	MEM 1 CONFIGURATION o 8K ROM o 16K ROM o 32K ROM o 64K ROM	*	2-7
W7	1-2 —	— 1-2	INTERRUPT VECTOR GENERATOR o Enable o Disable	*	2-7
W8	1-2 —	— 1-2	MEM 0 CONFIGURATION o 2K RAM o 8K RAM	*	2-6
W9	1-2 3-4	3-4 1-2	MEM 0 CONFIGURATION o 2K RAM o 8K RAM	*	2-6
W10	1-2,5-6 1-2,5-6 1-2,7-8 3-4,7-8	3-4,7-8 3-4,7-8 3-4,5-6 1-2,5-6	MEM 1 CONFIGURATION o 8K ROM o 16K ROM o 32K ROM o 64K ROM	*	2-7
W11 W12 W13	1-2,3-4 1-3,2-4	— --	RS-232-C CONFIGURATION o DCE o DTE	*	2-10
W14	1-2 3-4	3-4 1-2	TIMED INTERRUPT ENABLE o Enable o Disable	*	2-10
W15	1-2 3-4	3-4 1-2	BAUD RATE CLOCK SOURCE o Off-card RCLK (J1-8) o On-card Timer	*	2-11
W16	1-2 3-4	3-4 1-2	TIMED INTERRUPT CLOCK SOURCE o Timer 1 Output o Processor clock - 4	*	2-10
W17	— 3-4 5-6 3-4,5-6 1-2,5-6 1-2	1-2,3-4,5-6 1-2,5-6 1-2,3-4 1-2 3-4 3-4,5-6	RS-232-C SYNCHRONOUS CLOCK CONFIGURATION o No Clock o Transmit on RCLK o Transmit on TCLK o Transmit on RCLK and TCLK o Transmit on TCLK, Receive on RCLK o Receive on RCLK	*	2-11
W18	3-4 1-2	1-2 3-4	RS-232-C INTERRUPT ENABLE o Enable o Disable	*	2-12
W19	1-2 —	— 1-2	IOEXP CONFIGURATION o IOEXP Driven o IOEXP Not Connected	*	2-9
W20	1-2 —	— 1-2	MEMEX CONFIGURATION o MEMEX Driven o MEMEX Not Connected	*	2-9

Fig 2-4 cont'd

MEMORY CONFIGURATION

The 7862 has two memory sockets. MEM 0 is mapped with its starting address at 00000 hex. MEM 1 is mapped with its ending address at FFFFF hex. Either socket can be disabled by removing its appropriate jumper. If the socket is disabled, the 7862 can access off-card memory that is in the same address range as the disabled socket. See figures 2-5 and 2-6 for memory configuration jumpers.

The 7862 memory sockets accept byte-wide JEDEC standard ROMs and RAMs with an access time of 400 nanoseconds or less.

If the 4K ROM (2732A) is used as MEM 0, it is mapped either as a 2K or an 8K part. If mapped as a 2K part, only the lower half of the 4K ROM is useable. If mapped as an 8K part, the 4K ROM locations are redundantly mapped at 00000-00FFF and 01000-01FFF hex.

		W4		W8		W9	
MEMORY	ADDRESS RANGE	INSTALL	REMOVE	INSTALL	REMOVE	INSTALL	REMOVE
None	—	—	(1-2)	—	—	—	—
2K RAM*	00000-007FF	(1-2)	—	(1-2)	—	(1-2)	(3-4)
8K RAM	00000-01FFF	(1-2)	—	—	(1-2)	(3-4)	(1-2)
4K ROM	00000-007FF	(1-2)	—	(1-2)	—	(3-4)	(1-2)
4K ROM	00000-01FFF	(1-2)	—	—	(1-2)	(3-4)	(1-2)

*As shipped.

Figure 2-5. Socket 0 Jumper Configuration

		W5		W6		W10	
MEMORY	ADDRESS RANGE	INSTALL	REMOVE	INSTALL	REMOVE	INSTALL	REMOVE
None	——	—	(1-2)	—	—	—	—
8K ROM* (2764)	FE000-FFFFF	— (1-2) —	— — —	(1-2) (3-4) (5-6)	— — —	(1-2) (5-6) —	(3-4) (7-8) —
16K ROM (27158)	FC000-FFFFF	— (1-2)	— —	(1-2) (3-4)	— (5-6)	(1-2) (5-6)	(3-4) (7-8)
32K ROM (27256)	F8000-FFFFF	— (1-2)	— —	— (1-2)	(3-4) (5-6)	(1-2) (7-8)	(3-4) (5-6)
64K ROM (27512)	F0000-FFFFF	— (1-2) —	— — —	— — —	(1-2) (3-4) (5-6)	(3-4) (7-8)	(1-2) (5-6)

*As shipped.

Figure 2-6. Socket 1 Jumper Configuration

INTERRUPT VECTOR GENERATION

The 8088 CPU requires a single-byte interrupt vector during an interrupt acknowledge cycle. The 7862 provides an onboard vector generator for interrupting devices that are not able to supply their own vector. The jumper options allow enabling and disabling of the vector generator as well as the selection of one of four vectors. The interrupt vector generator must be disabled if any other device in the system is supplying interrupt vectors. The options are shown in figure 2-7.

DESCRIPTION	W2 JUMPERS (INSTALLED)	W7 JUMPERS (INSTALLED)
Disable Vector Generation	—	None
*Vector 05H	1-2,3-4	1-2
Vector 25H	1-2	1-2
Vector 45H	3-4	1-2
Vector 65H	None	1-2

*As shipped.

Figure 2-7. Interrupt Vector Generator Jumper Options (7862)

WAIT-STATE INSERTION

The 7862 supports the STD BUS WAITRQ* signal. This signal can be controlled by memory and I/O cards that cannot transfer information at the speeds required by the 7862. In addition to WAITRQ*, the 7862 can be jumper configured to insert a single wait state for access to all I/O devices. Figure 2-8 shows the jumper configuration for the wait-state insertion.

DESCRIPTION	W3 JUMPERS (INSTALLED)
*Enable I/O Wait State Insertion	1-2
Disable I/O Wait State Insertion	None

*As shipped.

Figure 2-8. I/O Wait State Insertion Jumper Configuration (7862)

WRITE TIMING SELECTION

The 7862 provides both an advanced WR* and a normal WR*. The write signal from the 8088 CPU becomes active early in the memory or I/O write cycle. The data being written by the CPU may not be valid at the time WR* goes active. This timing (advanced WR*) is not a problem for static memory or I/O cards because data is written on the low-to-high transition of WR*. Nonstatic memory systems, such as dynamic RAMs, may require a data setup time prior to the high-to-low transition of WR* (normal WR*). For data setup and hold times for both signals, refer to the 7862 Bus Write Timing specifications, figure 2-20, page 2-16. Figure 2-9 shows the jumper configurations used to select the advanced WR* or the normal WR* signal.

DESCRIPTION	W1 JUMPERS (INSTALLED)
**Normal WR*	1-2
Advanced WR*	3-4

**As shipped.

Figure 2-9. Write Timing Jumper Selection (7862)

MEMEX AND IOEXP CONFIGURATION

MEMEX and IOEXP on the 7862 are used to expand the memory and I/O capacity of the system. As shipped, the 7862 drives MEMEX with A19 and IOEXP with A15. If another card in the system drives MEMEX and/or IOEXP, the appropriate wire jumper(s) should be cut.

DESCRIPTION	W20 JUMPERS (INSTALLED)
*MEMEX driven	1-2
MEMEX disabled	Cut jumper

*As shipped.

Figure 2-10. MEMEX Configuration (7862)

DESCRIPTION	W19 JUMPERS (INSTALLED)
*IOEXP driven	1-2
IOEXP disabled	Cut jumper

*As shipped.

Figure 2-11. IOEXP Configuration (7862)

TIMER CONFIGURATION

The 7862 uses a programmable timer for various onboard timing functions. The programmable timer has three independent 16-bit timers. Timer 0 is used for baud rate generation. Timer 1 can be used for software delays or cascaded into timer 2. Timer 2 is used to generate timed interrupts. No jumper configuration is necessary for timers 0 and 1. The timer 2 clock source is jumper selectable from either a crystal controlled clock (the processor clock divided by four) or the output of timer 1. See figure 2-12 to configure the timer 2 clock source. The timer 2 interrupts should be disabled if the timer 2 is not used. See figure 2-13 to enable or disable the timer 2 interrupts.

DESCRIPTION	W16 JUMPERS (INSTALLED)
Timer 1 Output	1-2
*Processor Clock ÷ 4	3-4

*As shipped.

Figure 2-12. Timer 2 Clock Source (7862)

DESCRIPTION	W14 JUMPERS (INSTALLED)
Interrupts Enabled	1-2
*Interrupts Disabled	3-4

*As shipped.

Figure 2-13. Timer 2 Interrupts (7862)

SERIAL PORT CONFIGURATION

The 7862 has an RS-232-C serial communications port that can be configured as either DCE (Data Communications Equipment or modem) or DTE (Data Terminal Equipment or terminal). When operating the communications port as DCE, jumpers W11, 12, and 13 should be jumpered across pins 1-2 and pins 3-4. To operate the communications port as DTE, turn the six jumpers sideways and jumper pins 1-3 and pins 2-4. See figure 2-14 for configuration of the serial communications port.

DESCRIPTION	JUMPERS W11, 12, 13 (INSTALLED)
*DCE	1-2, 3-4
DTE	1-3,2-4

*As shipped.

Figure 2-14. Serial Port Configuration

The serial communications port on the 7862 operates either asynchronously, or synchronously using the BI-SYNC protocol.

In asynchronous mode, both serial ports operate with separate baud rate clocks. No baud rate clock is sent or received through the serial port. At least one start and stop bit is added to each byte transmitted to synchronize the communications. In this mode, timer 0 must be used as the baud rate clock generator.

In synchronous mode, the two serial ports share the same baud rate clock. This is done by transmitting the clock through the RS-232-C link. In a typical application, one serial port transmits its clock on the RCLK pin while the other serial port receives the clock. Some serial ports require two clock signals — one for receiving and one for transmitting data. For this special case, the 7862 can be jumper configured to transmit the baud rate clock on both the RCLK and TCLK pins.

Figure 2-15 shows the jumper configuration of the baud rate clock.

DESCRIPTION	W15	W17
*Asynchronous Mode	3-4	None
Transmit Clock on RCLK	3-4	3-4
Receive Clock on RCLK	1-2	1-2
Transmit Clock on RCLK and TCLK	3-4	3-4,5-6

*As shipped.

Figure 2-15. Baud Rate Clock Selection (7862)

The serial port can be either polled or interrupt driven. See figure 2-16 for jumper configuration. If the serial port interrupts are to be used, the onboard interrupt vector generator must be enabled and all possible interrupting devices must be polled in the interrupt service routine to determine the source of the interrupt.

DESCRIPTION	W18 JUMPERS (INSTALLED)
Interrupts Enabled	3-4
*Interrupts Disabled	1-2

*As shipped.

Figure 2-16. Serial Port Interrupt Configuration (7862)

7862 STD BUS INTERFACE

The 7862 is designed for use with the STD BUS system. The electrical signals defined by the STD BUS are listed in figure 2-17. Those shown with output drive or input load capabilities are the STD BUS interface signals used by the 7862.

The timing for the interface signals is shown in figures 2-18 through 2-23, pages 2-14 through 2-19. The timing has been divided into functional groups: Bus Status Timing (figure 2-18), Bus Read Timing (figure 2-19), Bus Write Timing (figure 2-20), Bus Interrupt Timing (figure 2-21), Bus Wait Timing (figure 2-22), and Bus Exchange Timing (figure 2-23).

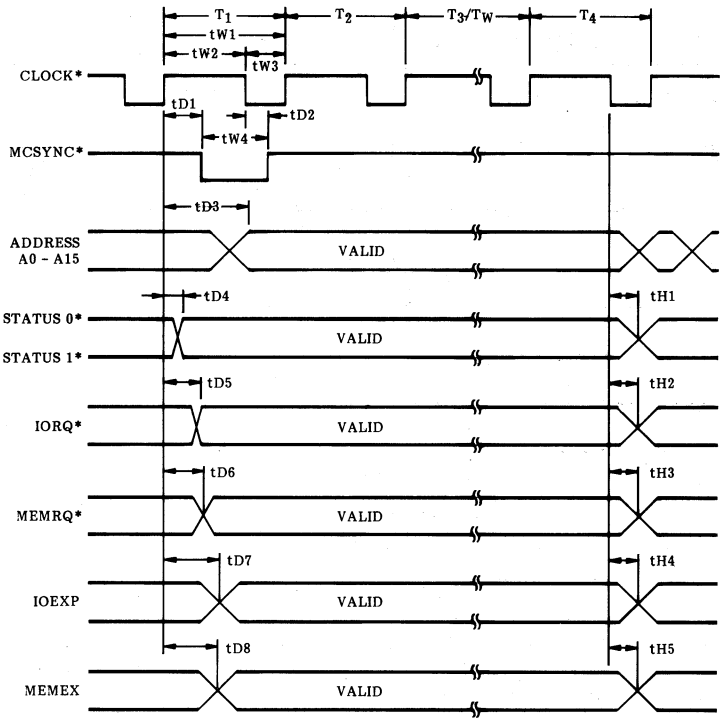
All times shown in figures 2-18 through 2-23 have been derated to include a temperature range of 0 to 65°C, a supply range of 4.75V to 5.25V DC, and a 45 pF load.

PIN NUMBER	OUTPUT (LSTTL DRIVE)	INPUT (LSTTL LOADS)	MNEMONIC		PIN NUMBER	OUTPUT (LSTTL DRIVE)	INPUT (LSTTL LOADS)	MNEMONIC
+5 VOLTS	IN		2		1		IN	+5 VOLTS
GROUND	IN		4		3		IN	GROUND
-5 VOLTS			6		5			-5 VOLTS
D7	1	55	8		7	55	1	D3/A19
D6	1	55	10		9	55	1	D2/A18
D5	1	55	12		11	55	1	D1/A17
D4	1	55	14		13	55	1	D0/A16
A15		55	16		15	55		A7
A14		55	18		17	55		A6
A13		55	20		19	55		A5
A12		55	22		21	55		A4
A11		55	24		23	55		A3
A10		55	26		25	55		A2
A9		55	28		27	55		A1
A8		55	30		29	55		A0
RD*		55	32		31	55		WR*
MEMRQ*		55	34		33	55		IORQ*
MEMEX		55	36		35	55		IOEXP
MCSYNC*		55	38		37			REFRESH*
STATUS 0*		55	40		39	55		STATUS 1*
BUSRQ*	30		42		41	55		BUSAK*
INTRQ*	30		44		43	55		INTAK*
NMIRQ	30		46		45		30	WAITRQ*
PBRESET*	(1)		48		47	55		SYSRESET*
CNTRL*			50		49	55		CLOCK*
PCI	IN		52		51	OUT		PCO
AUX GND			54		53			AUX GND
AUX -V	IN		56		55		IN	AUX +V

* Low-level active logic

(1) Switch Input

Figure 2-17. STD Edge Connector Pin List (7862)



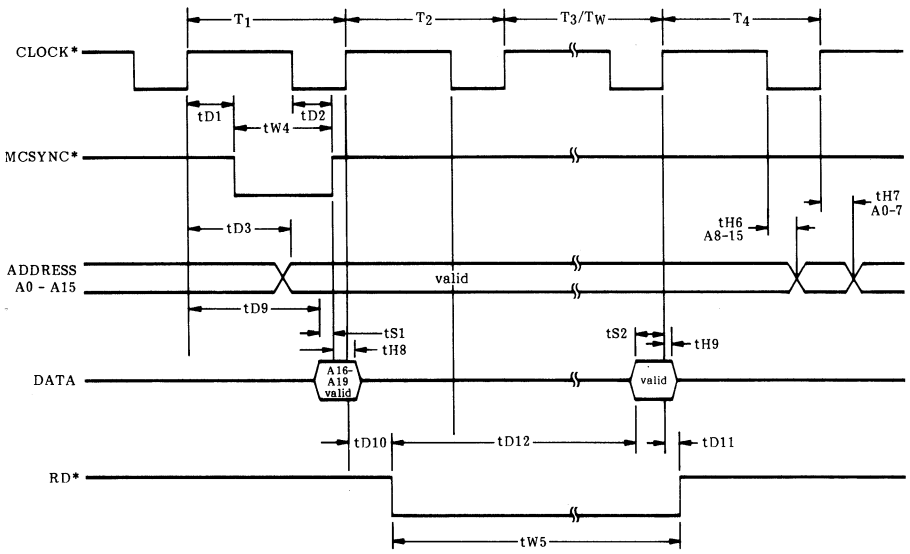
		4.915 MHz		
SYMBOL	**DESCRIPTION**	**MIN**	**TYP**	**MAX**
tW1	CLOCK* width[1]		203	
tW2	CLOCK* high width	116	127	138
tW3	CLOCK* low width	65	76	88

[1] Clock accuracy over temperature range is ±0.01%.

		4.915 MHz	
SYMBOL	**DESCRIPTION**	**MIN**	**MAX**
tD1	Delay from CLOCK* to MCSYNC* low	0	
tD2	Delay from CLOCK* to MCSYNC* high		107
tD3	Delay from CLOCK* to Address 0-15		152
tD4	Delay from CLOCK* to STATUS*		46
tD5	Delay from CLOCK* to IORQ*		64
tD6	Delay from CLOCK* to MEMRQ*		46
tD7	Delay from CLOCK* to IOEXP		129
tD8	Delay from CLOCK* to MEMEX		152
tH1	STATUS* hold after CLOCK*	0	
tH2	IORQ* hold after CLOCK*	0	
tH3	MEMRQ* hold after CLOCK*	0	
tH4	IOEXP hold after CLOCK*	0	
tH5	MEMEX hold after CLOCK*	62	
tW4	MCSYNC* pulse width	91	

All times given in nanoseconds.

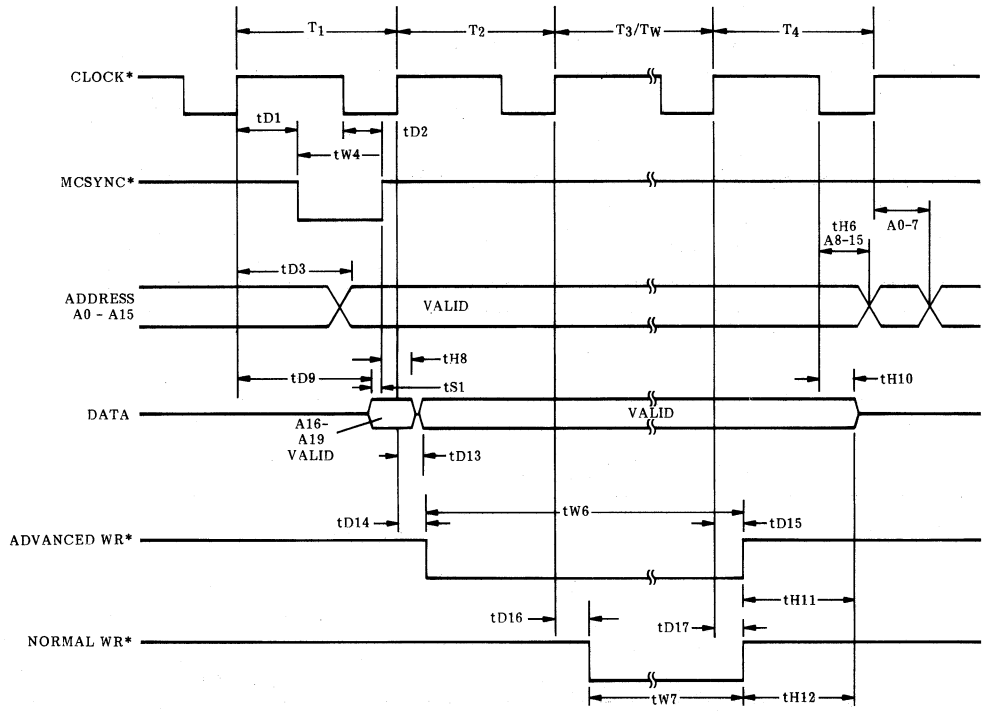
Figure 2-18. Bus Status Timing (7862)



		4.915 MHZ	
SYMBOL	**DESCRIPTION**	**MIN**	**MAX**
tD1	Delay from CLOCK* to MCSYNC* low	0	
tD2	Delay from CLOCK* to MCSYNC* high		107
tD3	Delay from CLOCK* to Address 0-15		152
tD9	Delay from CLOCK* to Address 16-19	0	131
tD10	Delay from CLOCK* to RD* low	0	171
tD11	Delay from CLOCK* to RD* high	0	156
tD12	Delay from RD* to Data Valid		186
tH6	Address 8-15 hold after CLOCK*	0	
tH7	Address 0-7 hold after CLOCK*	0	
tH8	Address 16-19 hold after MCSYNC*	21	
tH9	Data hold after CLOCK*	2	
tS1	Address 16-19 setup to MCSYNC*	45	
tS2	Data setup to CLOCK*	64	
tW4	MCSYNC* pulse width	91	
tW5	RD* pulse width	327	

All times given in nanoseconds.

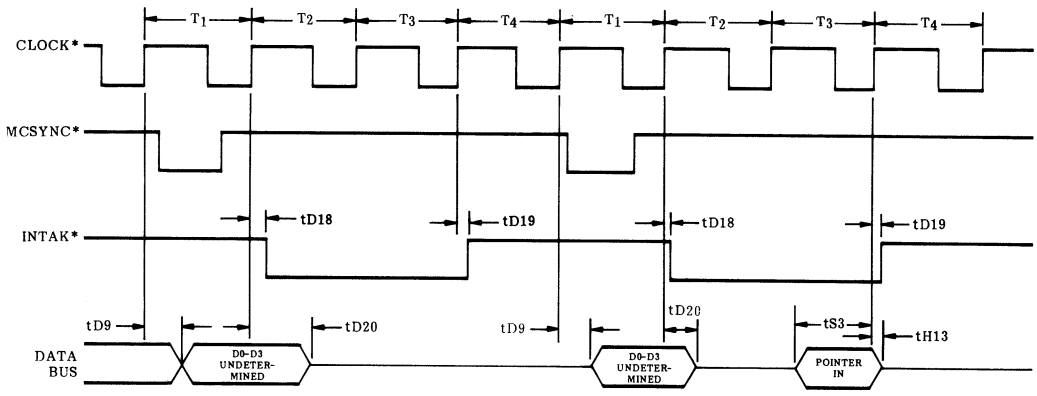
Figure 2-19. Bus Read Timing (7862)



		4.915 MHZ	
SYMBOL	**DESCRIPTION**	**MIN**	**MAX**
tD1	Delay from CLOCK* to MCSYNC* low	0	
tD2	Delay from CLOCK* to MCSYNC* high		107
tD3	Delay from CLOCK* to Address 0-15		152
tD9	Delay from CLOCK* to Address 16-19	0	131
tD13	Delay from CLOCK* to Data Valid		122
tD14	Delay from CLOCK* to Advanced WR* low	0	116
tD15	Delay from CLOCK* to Advanced WR* high	0	116
tD16	Delay from CLOCK* to Normal WR* low	0	73
tD17	Delay from CLOCK* to Normal WR* high	4	152
tH6	Address 8-15 hold after CLOCK*	0	
tH7	Address 0-7 hold after CLOCK*	0	
tH8	Address 16-19 hold after MCSYNC*	21	
tH10	Data hold after CLOCK*	0	
tH11	Data hold advanced after WR*	69	
tH12	Data hold after normal WR*	40	
tS1	Address 16-19 setup to MCSYNC*	45	
tW4	MCSYNC* pulse width	91	
tW6	Advanced WR* pulse width	342	
tW7	Normal WR* pulse width	167	

All times given in nanoseconds.

Figure 2-20. Bus Write Timing (7862)



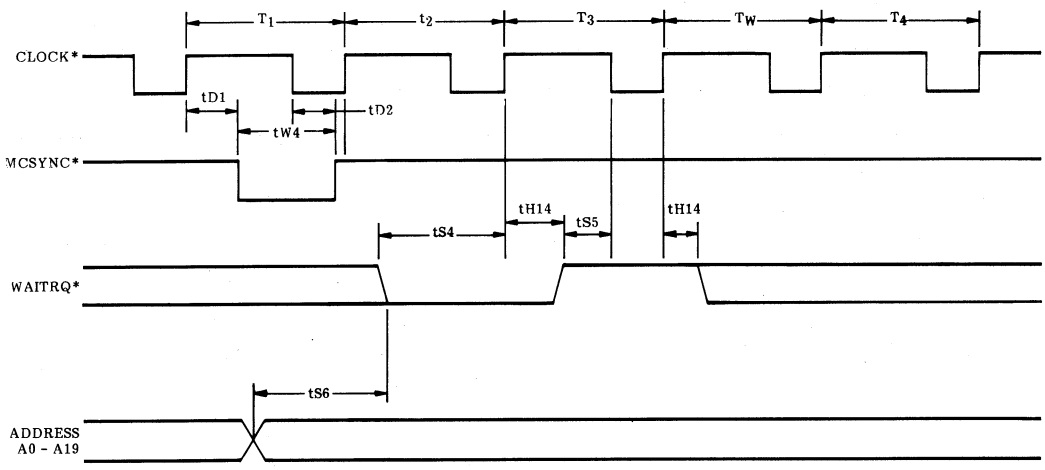
		4.915 MHZ	
SYMBOL	**DESCRIPTION**	**MIN**	**MAX**
tD9	Delay from CLOCK* to Address 16-19	0	131
tD18	Delay from CLOCK* to INTAK* low	0	116
tD19	Delay from CLOCK* to INTAK* high	0	116
tD20	Delay from CLOCK* to Data Bus 3-state		129
tH13	Interrupt vector hold from CLOCK*	2	
tS3	Interrupt vector setup to CLOCK*	64	

All times given in nanoseconds.

NOTES:

1. The illustrated example does not show possible additional clock cycles between T4 of first cycle and T1 of second INTAK cycle. These clock cycles occur if the 8088 internal queue is not full. These clock cycles do not affect system operation, only total interrupt response time.

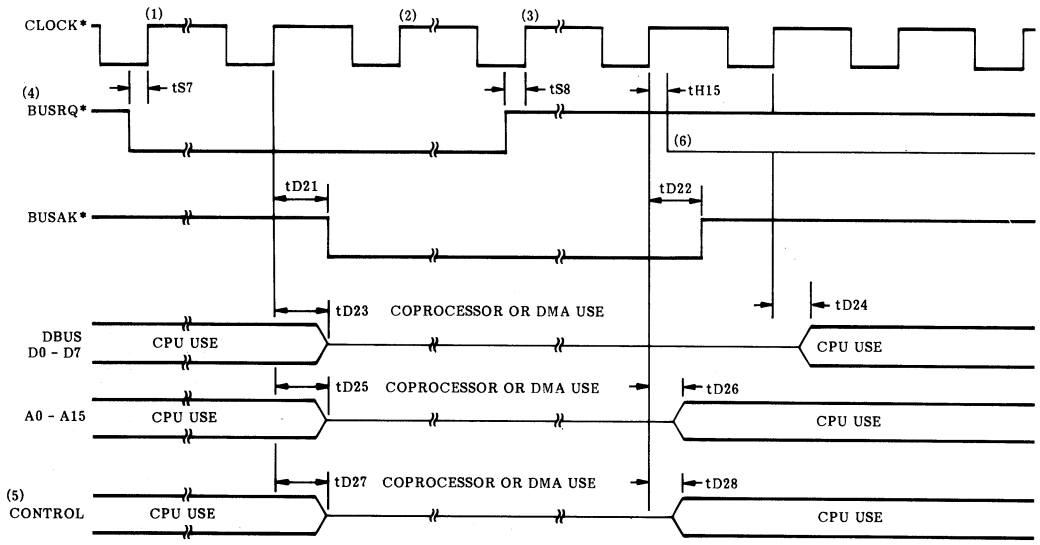
Figure 2-21. Bus Interrupt Timing (7862)



		4.915 MHZ	
SYMBOL	**DESCRIPTION**	**MIN**	**MAX**
tD1	Delay from CLOCK* to MCSYNC* low	0	
tD2	Delay from CLOCK* to MCSYNC* high		107
tH14	WAITRQ* hold after CLOCK*	0	
tS4	WAITRQ* low setup to COCK*	53	
tS5	WAITRQ* high setup to WAITRQ*	58	
tS6	Address 0-19 setup to WAITRQ*	216	
tW4	MCSYNC* pulse width	91	

All times given in nanoseconds.

Figure 2-22. Bus Wait Request Timing (7862)



		4.915 MHZ	
SYMBOL	**DESCRIPTION**	**MIN**	**MAX**
tD21	Delay from CLOCK* to BUSAK* low		165
tD22	Delay from CLOCK* to BUSAK* high		165
tD23	Delay from CLOCK* to Data Bus 3-state		143
tD24	Delay from CLOCK* to Data Bus driven	197	
tD25	Delay from CLOCK* to Addr Bus 3-state		187
tD26	Delay from CLOCK* to Addr Bus driven	193	
tD27	Delay from CLOCK* to Cntrl Bus 3-state		169
tD28	Delay from CLOCK* to Cntrl Bus driven	176	
tH15	BUSRQ* hold after CLOCK*	15	
tS7	BUSRQ* low setup to CLOCK*	52	
tS8	BUSRQ* high setup to CLOCK*	57	

All times given in nanoseconds.

NOTES:

1. Minimum clock delay is two for the 7862. More clocks may be required for completion of current CPU instruction.
2. Multiple clock delay for STD BUS use by coprocessor or DMA device.
3. Two or three clock delay for the 7862.
4. Timings shown are for an asynchronous BUSRQ*.
5. Control includes STD BUS lines MCSYNC*, IORQ*, MEMRQ*, RD*, WR*, STATUS0*, and STATUS1*.
6. The next BUSRQ* may optionally be asserted at this time or later.

Figure 2-23. Bus Exchange Timing (7862)

7862 USER INTERFACE

The 7862 user interface is a 26-pin connector labeled J1. This connector is the RS-232-C serial port interface. All signals on this connector are at RS-232-C levels (nominally +9.5V). Figure 2-24 shows which pins are inputs and which pins are outputs for $\overline{\text{D}}$CE and DTE configurations.

PIN NUMBER							
OUTPUT (LSTTL DRIVE)							
INPUT (LSTTL LOADS)							
MNEMONIC							MNEMONIC
			2	1			
TCLK		OUT	4	3	DTE	DCE	TD*
			6	5	DCE	DTE	RD*
RCLK	IN	OUT	8	7	DTE	DCE	RTS
			10	9	DCE	DTE	CTS
			12	11	DCE	DTE	DSR
DTR	DCE	DTE	14	13	OUT		GND
			16	15			
			18	17			
			20	19			
			22	21			
			24	23			

Figure 2-24. J1 Connector Pinout (7862)

A Pro-Log RC701-3 or RC702-3 interconnect cable can convert the 26-pin connector (J1) to a 25-pin D connector. See figure 2-25 for the wire list for this cable.

D CONNECTOR PIN	J1 CONNECTOR PIN	SIGNAL MNEMONIC
1	1	
2	3	TD*
3	5	RD*
4	7	RTS
5	9	CTS
6	11	DSR
7	13	GND
8	15	
9	17	
10	19	
11	21	
12	23	
13	25	
14	2	
15	4	TCLK
16	6	
17	8	RCLK
18	10	
19	12	
20	14	DTR
21	16	
22	18	
23	20	
24	22	
25	24	
N/C	26	

Figure 2-25. Wire List for J1 to 25-Pin D Connector Cable (7862)

SPECIFICATIONS

Electrical Specifications

		RECOMMENDED OPERATING LIMITS			ABSOLUTE NONOPERATING LIMITS		
MNEM.	PARAMETER	MIN	TYP	MAX	MIN	MAX	UNITS
V_{CC}	Logic supply	4.75	5.00	5.25	0	5.50	Volts
	Aux +V	11.40	12.00	12.60	0	15.00	Volts
	Aux -V	-12.60	-12.00	-11.40	-15.00	0	Volts

Electrical Specifications Over Operating Range

MNEM.	PARAMETER	MIN	TYP	MAX	UNITS
I_{CC}[1]	VCC Supply Current	—	650	1400	milliamps
	Aux +V Supply Current	—	15	30	milliamps
	Aux -V Supply Current	—	15	27	milliamps
	STD BUS Input Load	See Pin List.			
	STD BUS Output Load	See Pin List.			

[1] Without onboard memory.

Environmental Specifications

		RECOMMENDED OPERATING LIMITS			ABSOLUTE NONOPERATING LIMITS		
MNEM.	PARAMETER	MIN	TYP	MAX	MIN	MAX	UNITS
T_A	Free Air Operating Temp.	0	+25	+65	-40	+75	°C
R_H	Humidity (noncondensing)	5	—		95	5	95

PARAMETER	MIN	TYP	MAX	UNITS
High-level output voltage[1]	8.0	9.5	AUX +V	Volts
Low-level output voltage[1]	AUX -V	-9.5	-8.0	Volts
High-level short-circuit current	6.0	10.0	12.0	mA
Low-level short-circuit current	-6.0	-10.0	-12.0	mA
High-level input threshold voltage			3.0	Volts
Low-level input threshold voltage	-3.0			Volts

[1] At termination resistances of 3.0K minimum.

Mechanical Specifications

The 7862 meets all the general STD BUS mechanical specifications.

INTRODUCTION

This section discusses the operation of the 7862 processor card and any necessary programming of the 7862 functions. These functions are: the reset circuit, the interrupt generator, the serial port, the timer, and the memory and I/O addressing. The 8088 CPU is discussed in appendix A and the 8088 instruction set in appendix B.

RESET

The 7862 resets the 8088 CPU upon entering a reset state. For information on the 8088 reset operation, refer to appendix A. The 7862 supports a power-on reset and a pushbutton reset.

Power-On Reset

The 7862 reset circuit uses a voltage comparator and a precision voltage reference to monitor the level of the VCC supply. During a power-on reset, the 8088 CPU is held in a reset state for a maximum of 90 milliseconds after VCC has reached approximately 4.75 VDC.

Pushbutton Reset

The 7862 is forced into a reset state with an active level on the STD BUS pushbutton reset (PBRESET*) signal. PBRESET* can be controlled by a switch or driven with an open collector device. PBRESET* is debounced and must remain low a minimum of 5 milliseconds to guarantee a reset.

INTERRUPTS

The 7862 supports two STD BUS signals used by devices to signal interrupts. The signals are NMIRQ* (nonmaskable interrupt request) and INTRQ* (interrupt request). Every interrupt is assigned a type number that identifies it to the CPU. The type number is a single byte value which is multiplied by four in the CPU to point to an interrupt table that contains the address of the interrupt service routine. The multiplication of the vector by four allows the physical address (Code Segment and Instruction Pointer) of the interrupt service routine to be stored in the interrupt table.

Interrupt requests on the NMIRQ* signal are predefined as type 2 vectors. The 7862 interrupt vector generator has no effect on the response of the 8088 CPU to NMIRQ*. NMIRQ* processing is shown in figure 3-1.

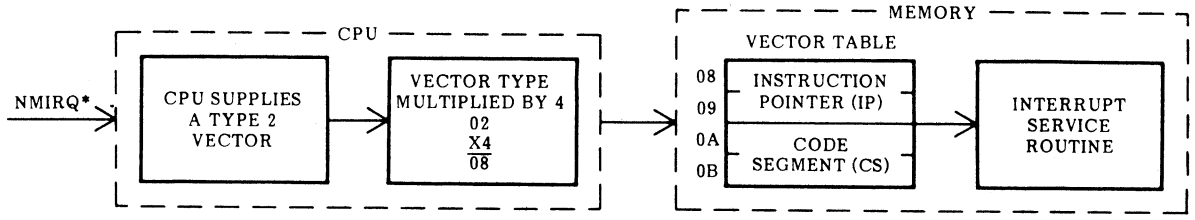


Figure 3-1. NMIRQ* Processing (7862)

Interrupt requests on the INTRQ* signal are not assigned a type vector by the CPU. The interrupting source must provide the vector type during the interrupt acknowledge cycle. INTRQ* processing is shown in figure 3-2.

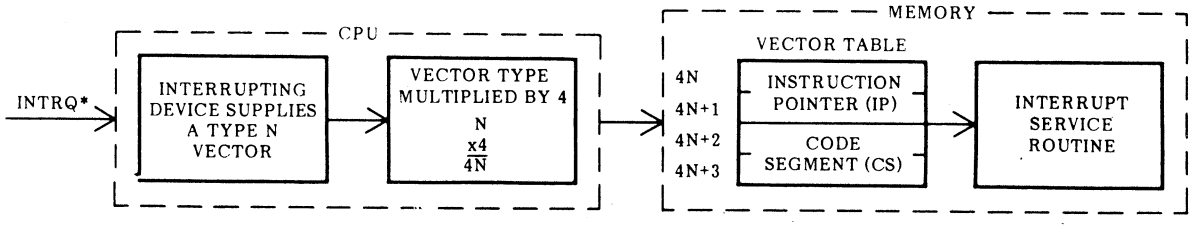


Figure 3-2. INTRQ* Processing (7862)

Many STD cards capable of generating an interrupt request are not capable of supplying a vector type. To be compatible with these cards, the 7862 has an onboard interrupt vector generator.

The 7862 interrupt vector generator provides the 8088 CPU with a jumper selectable vector during an interrupt acknowledge cycle. The vector points the CPU directly to a service routine if there is only one interrupt source driving the INTRQ* signal. If multiple interrupt sources are driving the INTRQ* signal, the vector can point the CPU to a software polling routine. The polling routine determines the source of the interrupt and points the CPU to that device's service routine. The Pro-Log 7320 Priority Interrupt Controller can provide a unique vector type for each source of interrupt.

The 7862 interrupt vector generator must be jumper disabled if the STD cards generating the interrupt requests are supplying their own vector. Refer to Card Layout and Jumpers, figure 2-2, page 2-2, for a listing of the available jumper options.

Bits D0 and D1 must be set to 00 for BI-SYNC or MONO-SYNC operation. For asynchronous mode, the baud rate factor can be either 1, 16, or 64. The rate factor causes the data to be transmitted and received on every clock, every sixteenth clock, or every sixty-fourth clock. If the 1X baud rate factor is used, both serial ports must share the same baud rate clock. Jumpers on the 7862 can select the baud rate clock to be either transmitted or received.

Bits D2 and D3 determine the number of data bits present in each data character.

Bits D4 and D5 determine the parity bit status. Select even, odd, or no parity. When even or odd parity is selected, the USART automatically inserts the correct polarity on transmit and checks for correct parity on receive.

In asynchronous mode, bits D6 and D7 determine the number of stop bits to be appended to each serial byte when transmitting; only one stop bit is required when receiving. In synchronous mode, bits D6 and D7 determine whether two sync characters (BI-SYNC) or one sync character (MONO-SYNC) is used to synchronize the serial ports.

COMMAND BYTE

The command byte is sent after the mode byte in asynchronous mode, or after the SYNC characters in synchronous mode. Subsequent command bytes can be sent at any time until the USART is reset. The format of the command byte is shown in figure 3-5.

D7	D6	D5	D4	D3	D2	D1	D0
EH	**SR**	**TH**	**ER**	**SBRK**	**REN**	**ERO**	**TEN**
Enter Hunt Mode	Software Reset	Transmit Handshake	Error Flag Reset	Send Break	Receive Enable	Equipment Ready Output	Transmit Enable
0 Inactive	0 Inactive	0 Inactive	0 Inactive	0 Inactive	0 Inactive	0 Inactive	0 Inactive
1 Momentary Active	1 Momentary Active	1 Active RTS for DTE, CTS for DCE)	1 Momentary Active	1 Active	1 Active	1 Active (DTR for DTE, DSR for DCE)	1 Active

Figure 3-5. USART Command Byte Format

Bit D0 (TEN) of the command word is the transmit enable bit. Data cannot be output from the USART unless this bit is in the active state. Transmit ready interrupts are disabled if this bit is cleared.

Bit D1 (ERO) of the command word is the equipment ready output bit. When the 7862 channel is configured as DTE, it sets the data terminal ready (DTR) signal. When the 7862 channel is configured as DCE, it sets the data set ready (DSR) signal. This bit and the transmit handshake bit control outgoing handshaking.

Bit D2 (REN) is the receive enable command bit. This bit enables the received ready flag of the status port. However, this bit does not prevent the assembly of serial characters. Receive ready interrupts are disabled if this bit is cleared.

Bit D3 (SBRK) is the send break command bit. When this bit is set, the serial transmitted data is interrupted and a continuous low level (spacing) signal is transmitted. This signal continues as long as this bit is set.

Bit D4 (ER) is the error reset bit. It clears the three error flags of the status port. This bit performs a momentary function.

Bit D5 (TH) is the transmit handshake bit. When the 7862 is configured as DTE, this bit sets the request to send (RTS) signal. When the 7862 is configured as DCE, it sets the clear to send (CTS) signal. This bit and the equipment ready output bit control outgoing handshaking.

Bit D6 (SR) is the software reset bit performing a momentary function. The USART channel enters the reset state.

Bit D7 (EH) is the enter hunt mode bit. It is used only in synchronous mode and should be 0 in asynchronous mode. This bit should be set in the first command byte following a synchronous mode byte. When this bit is set, the serial data is sampled until the SYNC character or characters are found. The SYNC detect bit is then set and the USART ends the hunt mode.

STATUS BYTE

The status byte always contains the present status of the USART and can be read by the processor at any time. The status byte can be obtained by reading from port address FFF9 hex. The format of the status byte is shown in figure 3-6.

Bit D0 (TRDY) is the transmit ready bit. It indicates that the transmit character buffer is empty and that the USART can accept a new character for transmission. This bit is cleared when the transmit buffer is loaded by the CPU. Bit 0 should be used to determine when to output the next data byte.

Bit D1 (RRDY) is the receive ready bit which indicates that the receive buffer has a character available. This bit is cleared when the receive buffer has been emptied by the CPU.

Bit D2 (IOE) indicates when the transmit output buffer is empty.

Bit D3 (PE) is the parity error signal which indicates that the received character has a parity error. This bit is reset by the error reset bit in the command word.

Bit D4 (OE) is the overrun error bit which indicates that the received character has been overwritten by the next character. This bit is reset by the error reset bit in the command word.

Bit D5 (FE) is the framing error bit which indicates that the received character has an incorrect number of bits or the wrong number of stop bits. This bit is reset by the error reset bit in the command word.

Bit D6 (BRK DET/SYNC DET) is the break detect bit in asynchronous mode. This bit goes high when the receiver remains low through two consecutive stop bit sequences, indicating a break in transmission. Bit 6 is the SYNC detect bit in synchronous mode. This bit goes high when the USART has located the SYNC character in receive mode. Sync detect is reset upon a status read operation.

Bit D7 (ERI) is the equipment ready input bit which indicates the status of the incoming handshake line. When the 7862 channel is configured as DTE, this bit indicates the state of the data set ready (DSR) line. When the 7862 channel is configured as DCE, it indicates the state of the data terminal ready (DTR) line.

D7	D6	D5	D4	D3	D2	D1	D0
ERI	BRK DET/ SYNC DET	FE	OE	PE	IOE	RRDY	TRDY
Equipment Ready Input	Break Detect/ Sync Detect	Framing Error	Overrun Error	Parity Error	Transmit Output Buffer Empty	Receive Ready Bit	Transmit Ready Bit
0 Inactive	0 Inactive	0 Inactive	0 Inactive	0 Inactive	0 Inactive	0 Inactive	0 Inactive
1 Active (DRS for DTE, DTR for DCE)	1 Active	1 Active	1 Active	1 Active	1 Active	1 Active	1 Active

Figure 3-6. USART Status Byte Format

PROGRAMMABLE INTERVAL TIMER

The programmable interval timer (8253) has three independent 16-bit timers. Each timer on the 7862 is dedicated to a specific function; therefore, only the valid timer modes for those functions are described.

Timer 0

This timer is used for onboard baud rate generation on the 7862. The timer must be initialized before the serial port is used unless the USART baud rate is received from off-board.

To initialize the baud rate clock, timer 0 must be initialized with the following bytes. See the table in figure 3-7 to find the Most Significant Byte (MSB) and the Least Significant Byte (LSB) of the count for the desired baud rate.

PORT	BYTE	DESCRIPTION
FFFF	36H	Square Wave Mode
FFFC	XXH	LSB of count
FFFC	XXH	MSB of count

BAUD RATE	COUNT		ACTUAL BAUD RATE	% ERROR
	MSB	LSB		
19200	00	04	19200	0
9600	00	08	9600	0
4800	00	10	4800	0
2400	00	20	2400	0
1200	00	40	1200	0
600	00	80	600	0
300	01	00	300	0
150	02	00	200	0
134.5	02	3B	134.50	0.026
110	02	BA	110.03	0.026
75	04	00	75	0
50	06	00	50	0

COUNTER VALUES FOR BAUD RATE FACTOR OF 16X

BAUD RATE	COUNT		ACTUAL BAUD RATE	% ERROR
	MSB	LSB		
61440*	00	14	61440	0
38400*	00	20	38400	0
19200	00	40	19200	0
9600	00	80	9600	0
4800	01	00	4800	0
2400	02	00	2400	0
1200	04	00	1200	0
600	08	00	600	0
300	10	00	300	0

NOTE: RS-232-C specifies a maximum data rate of therefore, these rates do not meet RS-232-C specifica applications, however, these data rates are acceptable.

COUNTER VALUES FOR BAUD RATE FACTOR OF 1X

Figure 3-7. Baud Rate Selection

Timer 1

This timer has two possible uses on the 7862. It is always connected to the 8088 TEST* pin to generate software delays, but it can be jumpered to the clock input of timer 2 to generate longer time intervals with timer 2.

To generate software delays, timer 1 must be initialized with the following bytes:

PORT	BYTE	DESCRIPTION
FFFF	70H	Mode 0 for Timer 1
FFFD	XXH	LSB of count
FFFD	XXH	MSB of count

To determine the count, divide the delay by 813.8 nanoseconds and convert to hexadecimal. After loading the mode byte, a delay can be generated by loading the count bytes and executing a WAIT instruction.

Timer 1 can also be used as a programmable clock for timer 2. To use this feature, jumper W16 must have pins 1-2 shorted and pins 3-4 open. Timer 1 must be initialized with the following bytes:

PORT	BYTE	DESCRIPTION
FFFF	76H	Mode 3 for Timer 1
FFFD	XXH	LSB of divisor
FFFD	XXH	MSB of divisor

To determine the divisor, divide the desired clock period for timer 2 by 813.8 nanoseconds.

Timer 2

This timer is used to generate periodic interrupts. Its clock input can come from either the 813.8 nanosecond clock or from the output of timer 1. To initialize this timer for interrupt generation, the following bytes must be sent:

PORT	BYTE	DESCRIPTION
FFFF	B0H	Mode 0 for Timer 2
FFFE	XXH	LSB of count
FFFE	XXH	MSB of count

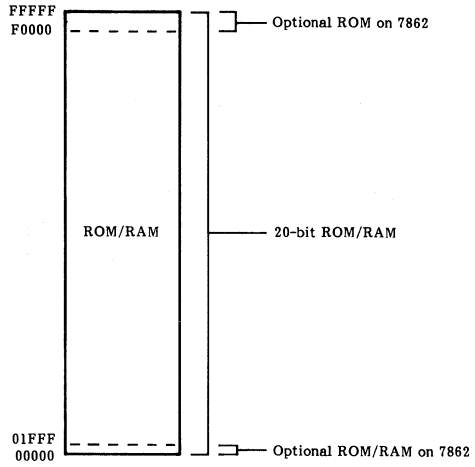
To determine the count, divide the period between interrupts by either 813.8 nanoseconds or the output clock period of timer 1. After the last count byte is sent, the timer counts down and generates an interrupt when the count goes to zero. The two-byte count must be reloaded to generate another interrupt.

SYSTEM MEMORY ADDRESSING

The 7862 provides a 20-bit memory address during each memory operation. The lower 16 bits are output on A0-A15. The upper four bits (A16-A19) are time multiplexed onto D0-D3 at the beginning of each memory cycle. Memory cards can latch A16-A19 on the rising edge of MCSYNC*. Many STD memory cards do not decode A16-A19; therefore, A19 is latched and output on MEMEX to increase the total possible system memory.

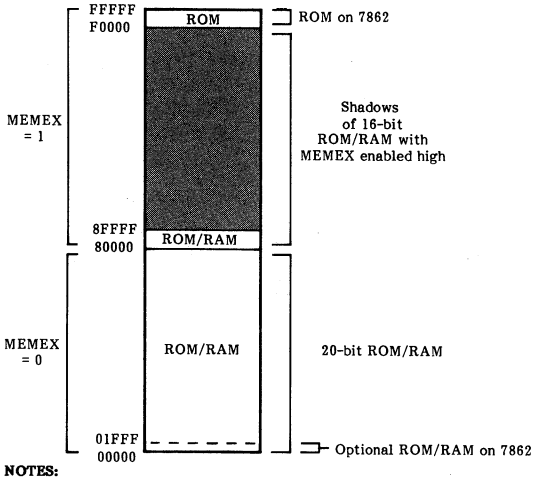
Some sample system memory configurations are shown in figure 3-8. These samples show the maximum memory (ROM and RAM) available for certain combinations of 16- and 20-bit memory cards. The samples should serve as a guide for configuring a particular system memory.

Shadowed memory occurs when the memory decoding hardware does not decode all of the address lines. Software can read or write to different "shadow" addresses of memory that are actually the same address in the physical memory.



NOTE: Maximum memory is 1 megabyte of ROM/RAM.

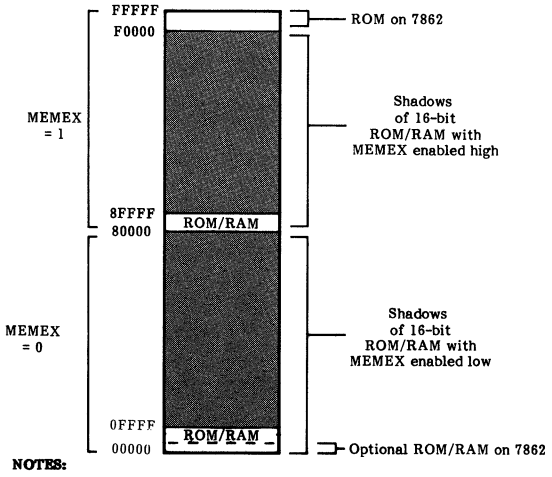
20-Bit Addressing



NOTES:

1. Maximum memory is 64K bytes of ROM; 576K bytes of RAM.
2. A memory write to ROM at address F0000-FFFFF will destroy data in RAM at address 80000-8FFFF.

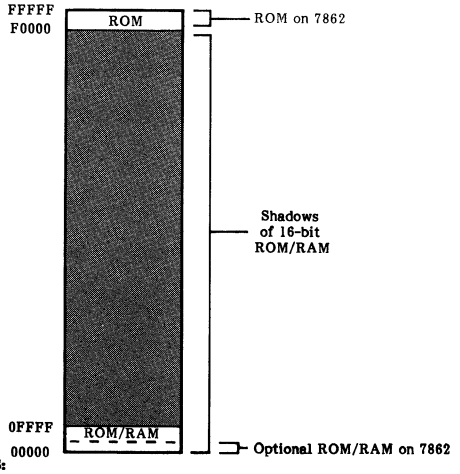
Mixed 20-Bit and 16-Bit Addressing With MEMEX



NOTES:

1. Maximum memory is 64K bytes of ROM; 128K bytes of ROM/RAM.
2. A memory write to ROM at address F0000-FFFFF will destroy data in RAM at address 80000-8FFFF.

16-Bit Addressing With MEMEX



NOTES:

1. Maximum memory is 64K bytes of ROM; 64K bytes of ROM/RAM.
2. A memory write to ROM at address F0000-FFFFF will destroy data in RAM at address 00000-0FFFF.

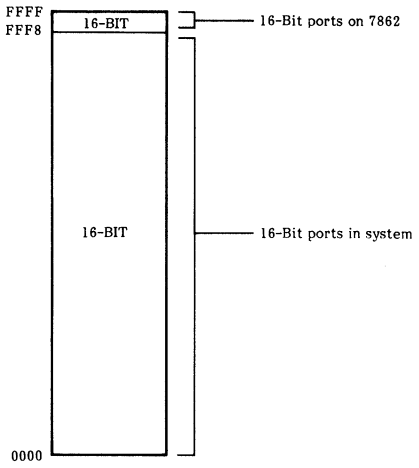
16-Bit Addressing Without MEMEX

Figure 3-8. Sample System Memory Configurations

SYSTEM I/O ADDRESSING

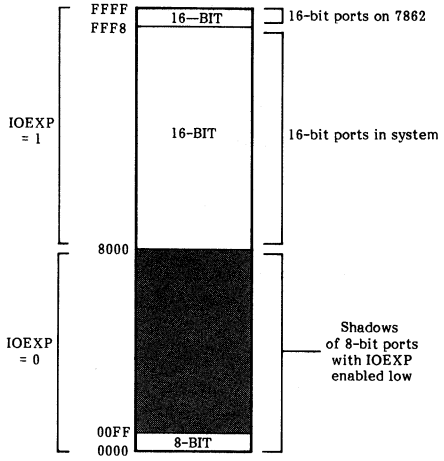
The 7862 provides a 16-bit I/O port address during each I/O operation. Many STD cards do not decode A8-A15; therefore, A15 is output on IOEXP to increase the total possible I/O ports available.

Some sample I/O port configurations are shown in figure 3-9. These samples show the maximum number of I/O ports available for certain combinations of 8- and 16-bit I/O port addressing cards. The samples should serve as a guide for configuring a particular system's I/O ports.



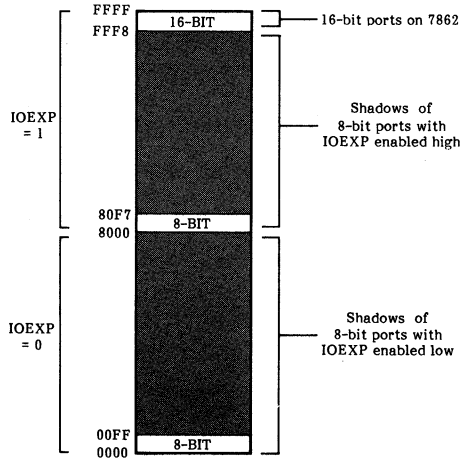
NOTE: Maximum I/O ports are 62,536 16-bit ports.

16-Bit Addressing



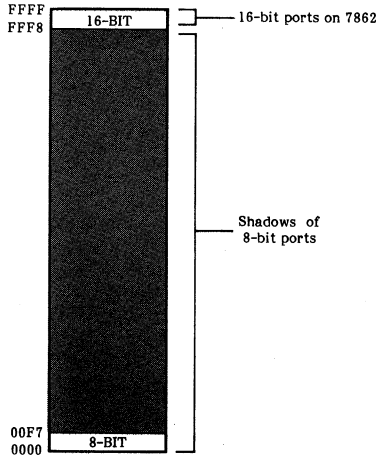
NOTE: Maximum I/O ports are 32,768 16-bit ports; 256 8-bit ports.

Mixed 16-Bit and 8-Bit Addressing With IOEXP



NOTE: Maximum I/O ports are 8 16-bit ports; 504 8-bit ports.

8-Bit Addressing With IOEXP



NOTE: Maximum I/O ports are 8 16-bit ports; 248 8-bit ports

8-Bit Addressing Without IOEXP

Figure 3-9. Sample SYSTEM I/O Port Configurations

INTRODUCTION

This section contains hardware level subroutine modules with which to operate the 7862.

The software in this section can be used without license from Pro-Log. Although tested and believed correct, this software is not represented to be free from errors or copyright infringement or appropriate for any specific application.

The subroutines are in STD instruction mnemonics using 8088 assembly codes. The coding forms are grouped at the end of this section following the flowcharts.

The subroutines are grouped in functional modules. Each module specification describes the module's content. Individual subroutine specifications give memory, entry, and exit requirements for each path, plus timing and other necessary information.

MEMORY ADDRESSES

Any addressing within the subroutines is relative addressing; therefore, the routines can be relocated without modification.

I/O PORT ADDRESSES

The 7862 I/O port addresses are fixed at address FFF8-FFFF hex.

SUBROUTINE: (INIT ASYNC COMM)

PROGRAM MEMORY	:	51 bytes
DATA MEMORY	:	0 bytes
STACK MEMORY	:	8 bytes
REGISTERS ALTERED	:	None
SUBOUTINES USED	:	None
ENTRY CONDITIONS	:	Call with JSR
EXIT CONDITIONS	:	None

This subroutine initializes the timer for 9600 baud then resets and initializes the USART for asynchronous communications with 8 data bits, 1 stop bit, and no parity.

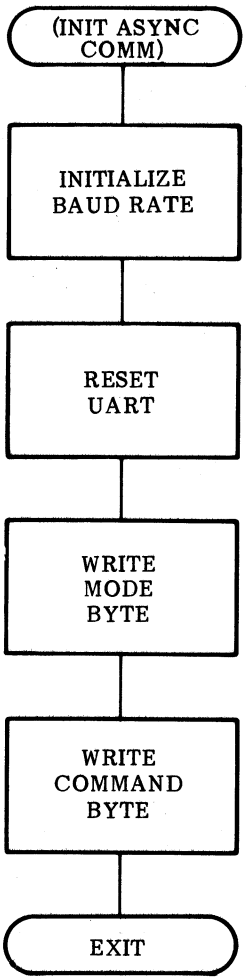


Figure 4-1. (INIT ASYNC COMM) Flowchart (7862)

SUBROUTINE: (INIT SYNC COMM)

PROGRAM MEMORY	:	72 bytes
DATA MEMORY	:	0 bytes
STACK MEMORY	:	8 bytes
REGISTERS ALTERED	:	None
SUBROUTINES USED	:	(CHAR OUT)
ENTRY CONDITIONS	:	Call with JSR
EXIT CONDITIONS	:	None

This subroutine initializes the timer for 9600 baud and resets and initializes the USART for synchronous communications with 8 data bits and no parity. The routine then synchronizes both communications ports. **NOTE:** The subroutine does not return until sync characters are received.

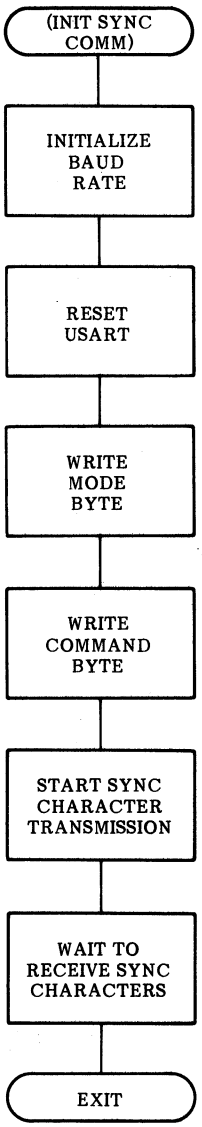


Figure 4-2. (INIT SYNC COMM) Flowchart (7862)

SUBROUTINE: (CHAR OUT)

PROGRAM MEMORY	:	21 bytes
DATA MEMORY	:	0 bytes
STACK MEMORY	:	8 bytes
REGISTERS ALTERED	:	None
SUBROUTINES USED	:	None
ENTRY CONDITIONS	:	Call with JSR A = Character
EXIT CONDITIONS	:	None

This subroutine outputs a character to the serial port.

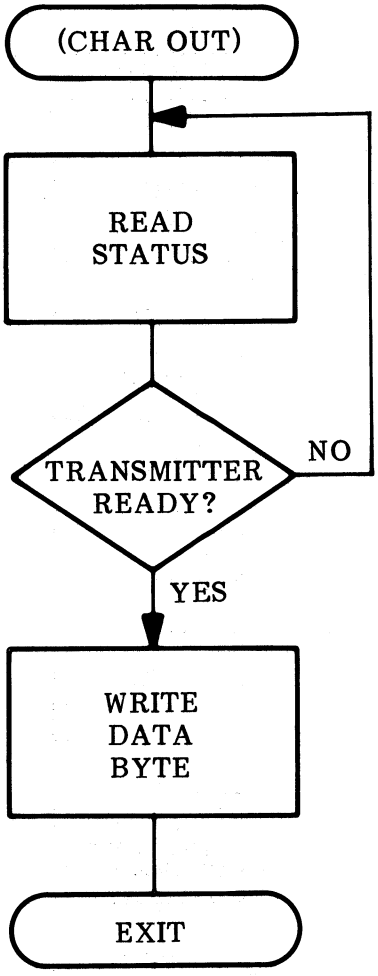


Figure 4-3. (CHAR OUT) Flowchart (7862)

SUBROUTINE: (CHAR IN)

PROGRAM MEMORY	:	20 bytes
DATA MEMORY	:	0 bytes
STACK MEMORY	:	4 bytes
REGISTERS ALTERED:	:	A, Flags
SUBROUTINES USED	:	None
ENTRY CONDITIONS	:	Call with JSR
EXIT CONDITIONS	:	A = Character Z = 0, error occurred Z = 1, no error

This subroutine inputs a character from the serial port.

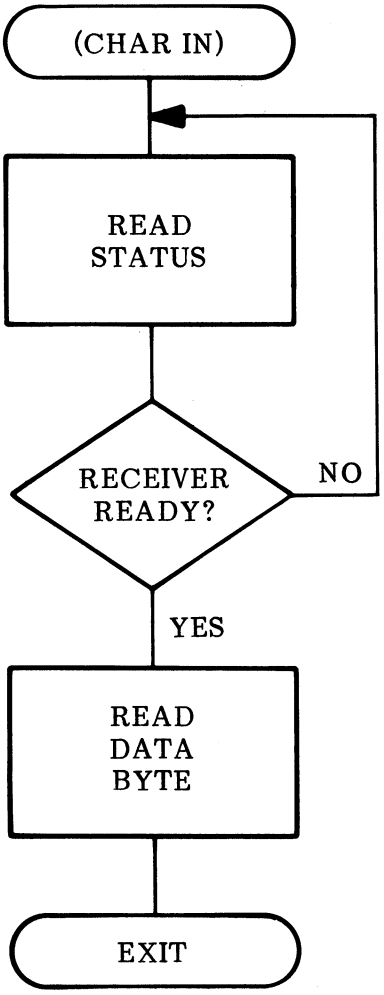


Figure 4-4. (CHAR IN) Flowchart (7862)

SUBROUTINE: (SYNC BLOCK OUT)

PROGRAM MEMORY	:	87 bytes
DATA MEMORY	:	Block Size +2
STACK MEMORY	:	20 bytes
REGISTERS ALTERED	:	None
SUBROUTINES USED	:	(CHAR OUT), (GEN CRC)
ENTRY CONDITIONS	:	Call with JSR DS = Segment of block HL = Offset of start of block BC = Number of bytes to output
EXIT CONDITIONS	:	None

This subroutine outputs a block of data from memory to the serial port. The data begins at (HL) in the data segment and is BC bytes long. The block is transmitted using the BI-SYNC transparent data protocol (see figure 4-6). This subroutine should be used only with baud rates at or below 9600 baud.

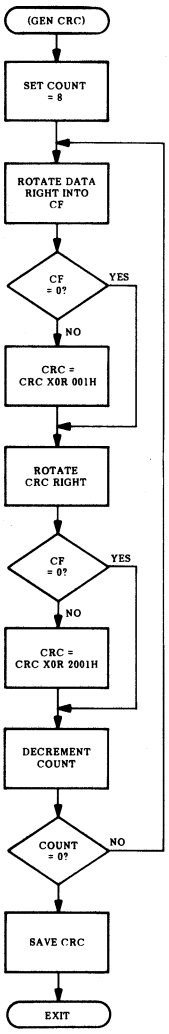


Figure 4-5. (SYNC BLOCK OUT) Flowchart (7862)

DLE	SYN	DLE	STX	Transparent Data*	DLE	ETX

*Any DLE data character is sent as two DLE characters.

Figure 4-6. BY-SYNC Protocol

SUBROUTINE: (SYNC BLOCK IN)

PROGRAM MEMORY	:	103 bytes
DATA MEMORY	:	Block Size +2
STACK MEMORY	:	16 bytes
REGISTERS ALTERED	:	A, B, C, Flags
SUBROUTINES USED	:	(CHAR IN), (GEN CRC)
ENTRY CONDITIONS	:	Call with JSR DS = Segment of block HL = Offset of start of block
EXIT CONDITIONS	:	BC = Number of bytes input Z = 0, error occurred Z = 1, no error

This subroutine inputs a block of data from the serial port and saves it in memory starting at address (HL) in the data segment. The BC register returns the number of bytes input. This subroutine should be used only with baud rates at or below 9600 baud.

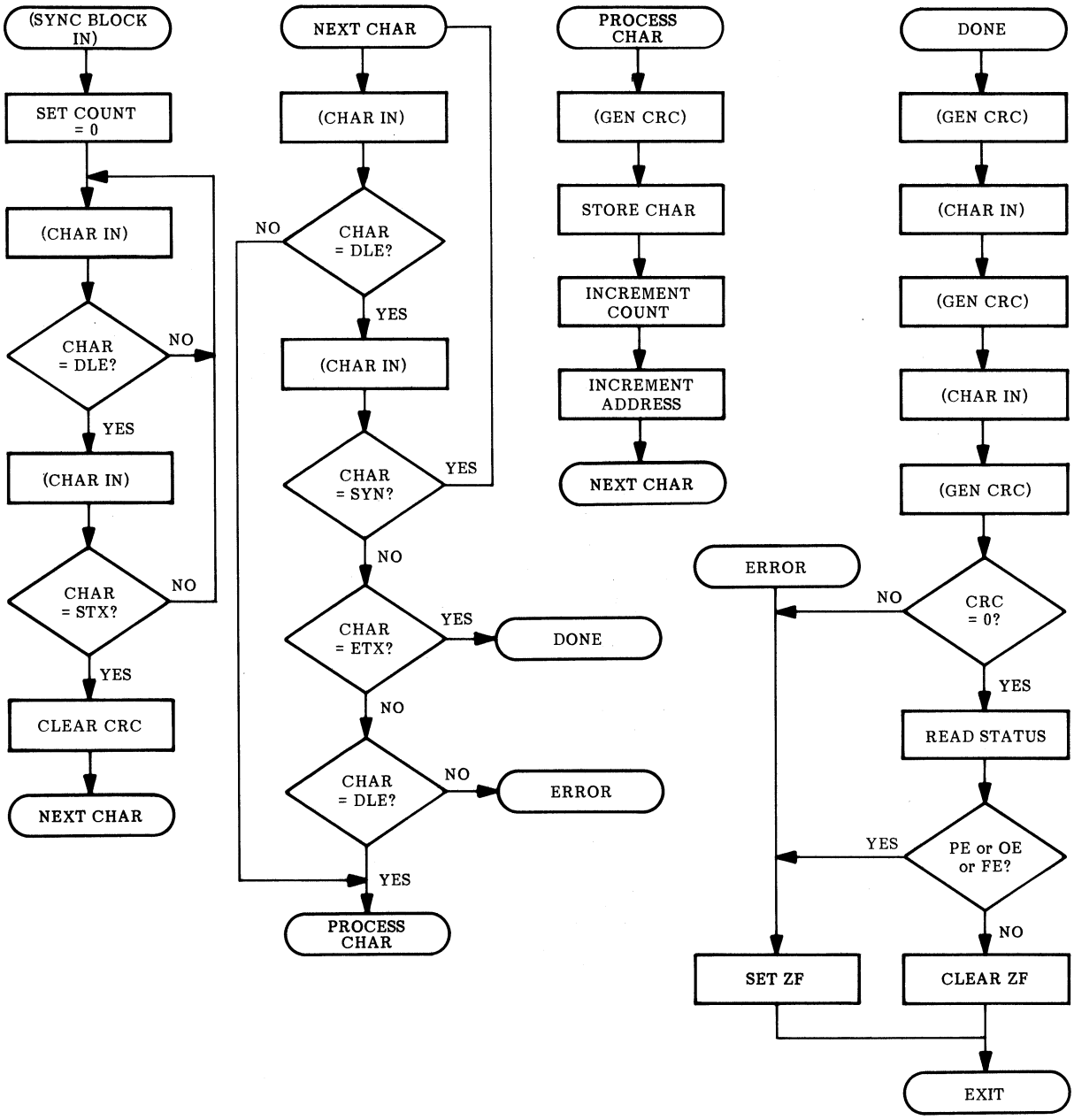


Figure 4-7. (SYNC BLOCK IN) Flowchart (7862)

SUBROUTINE: (GEN CRC)

PROGRAM MEMORY	:	87 bytes
DATA MEMORY	:	2 bytes
STACK MEMORY	:	10 bytes
REGISTERS ALTERED	:	None
SUBROUTINES USED	:	None
ENTRY CONDITIONS	:	Call with JSR A = Byte
EXIT CONDITIONS:	:	CRC word in memory is updated

This subroutine updates the CRC (Cyclic Redundancy Check) byte using the data byte in A as the next data. The subroutine uses an algorithm to generate a CRC-16 standard check character, which is stored in memory so it can be updated for the next data byte.

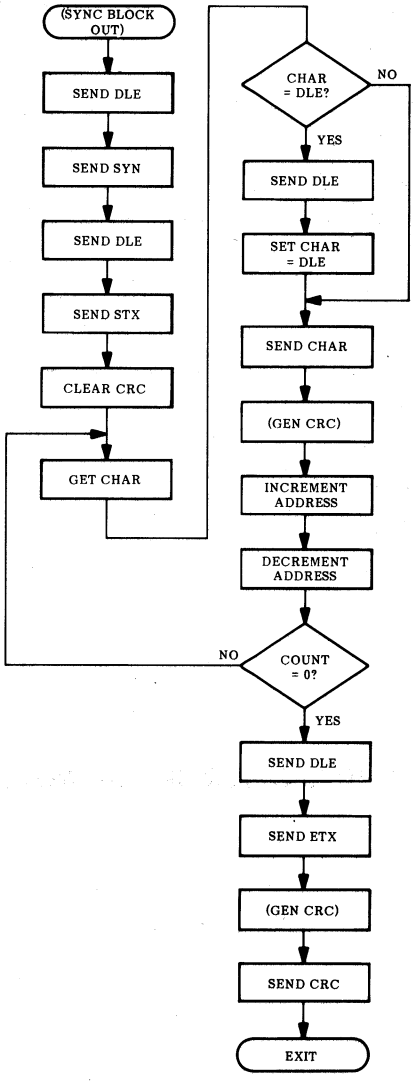


Figure 4-8. (GEN CRC) Flowchart (7862)

SUBROUTINE: (DELAY)

PROGRAM MEMORY	:	25 bytes
DATA MEMORY	:	0 bytes
STACK MEMORY	:	8 bytes
REGISTERS ALTERED	:	None
SUBROUTINES USED	:	None
ENTRY CONDITIONS	:	Call with JSR BC = Delay count
EXIT CONDITIONS	:	None

This subroutine uses timer 1 and 8088 TEST* pin to generate a delay. The delay time is approximately:

$$t_D = \frac{BC+98}{1.2288 \times 10^6}$$

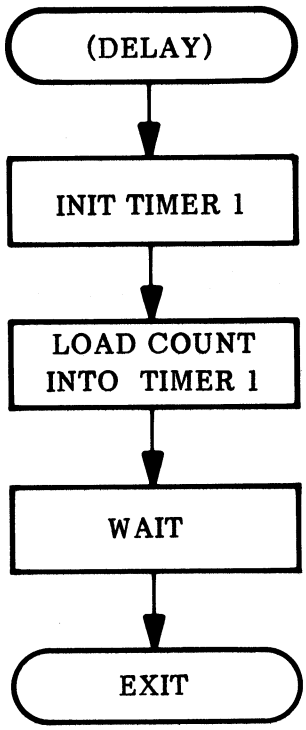


Figure 4-9. (DELAY) Flowchart (7862)

SUBROUTINE: (INIT INTR)

PROGRAM MEMORY	:	35 bytes
DATA MEMORY	:	4 bytes
STACK MEMORY	:	2 bytes
REGISTERS ALTERED	:	A,D,E,H,L
SUBROUTINES USED	:	None
ENTRY CONDITIONS	:	Call with JSR
EXIT CONDITIONS	:	None

This subroutine stores the interrupt routine (INTR) segment and offset at the appropriate vector address. The timer is then initialized and loaded and interrupts are enabled.

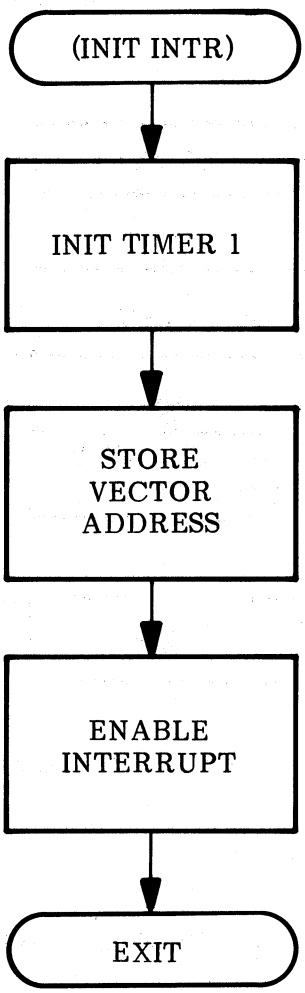


Figure 4-10. (INIT INTR) Flowchart

SUBROUTINE: (INTR)

PROGRAM MEMORY	:	32 bytes
DATA MEMORY	:	0 bytes
STACK MEMORY	:	10 bytes
REGISTERS ALTERED	:	None
SUBROUTINES USED	:	None
ENTRY CONDITIONS	:	None
EXIT CONDITIONS	:	None

This is the interrupt subroutine. On entry, all of the registers are saved and the timer is reloaded for the next interrupt. The user program is executed, then interrupts are enabled and the registers restored. The periodic time is approximately:

$$t_D = \frac{\text{DELAY} + 102}{1.2288 \times 10^6}$$, if Timer 2 input is 1.23 MHz (W16, pin 3-4)

$$t_D = \frac{\text{DELAY} \times (\text{Timer 2 Count}) + 103}{1.2288 \times 10^6}$$, if Timer 2 input is Timer 1 (W16, pin 1-2)

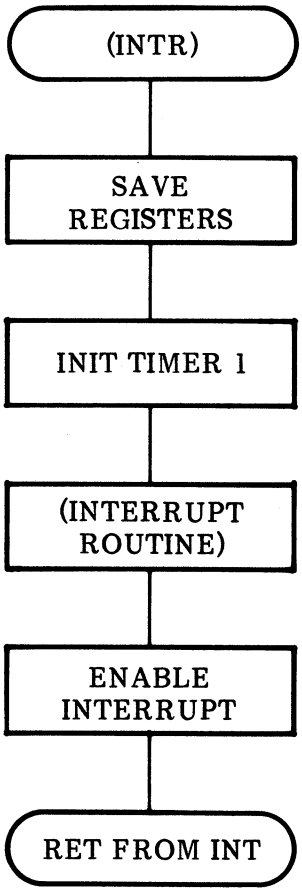


Figure 4-11. (INTR) Flowchart (7862)

```
ADRdata!     LABEL       !INSTR !     MODIFIER    !           COMMENTS
-------!----------------!------!-----------------!-----------------------------
1000 50!(INIT ASYNC COM)!PSP   !AF               !
  01 9C!                 ! ---- !                 !
  02 52!                 !PSP   !DE               !
  03 C7!                 !LDPI  !DE               !INITIALIZE BAUD RATE ...
  04 C2!                 ! ---- !                 !
  05 FF!                 ! ---- !FFFF             !
  06 FF!                 ! ---- !                 !
  07 B0!                 !LDAI  !                 !
  08 36!                 ! ---- !36               !
  09 EE!                 !OPAN  !(DE)             !SET TIMER 0 TO OUTPUT A SQUARE
  0A C7!                 !LDPI  !DE               !  WAVE
  0B C2!                 ! ---- !                 !
  0C FC!                 ! ---- !FFFC             !
  0D FF!                 ! ---- !                 !
  0E B0!                 !LDAI  !                 !BAUD RATE=9600
  0F 08!                 ! ---- !8                !LSB OF BAUD RATE DIVISOR
  10 EE!                 !OPAN  !(DE)             !
  11 B0!                 !LDAI  !                 !
  12 00!                 ! ---- !0                !MSB OF BAUD RATE DIVISOR
  13 EE!                 !OPAN  !(DE)             !
  14 C7!                 !LDPI  !DE               !RESET USART ...
  15 C2!                 ! ---- !                 !
  16 F9!                 ! ---- !FFF9             !
  17 FF!                 ! ---- !                 !
  18 B0!                 !LDAI  !                 !
  19 00!                 ! ---- !0                !
  1A EE!                 !OPAN  !(DE)             !
  1B B0!                 !LDAI  !                 !
  1C 00!                 ! ---- !0                !
  1D EE!                 !OPAN  !(DE)             !
  1E B0!                 !LDAI  !                 !
  1F 00!                 ! ---- !0                !
1020 EE!                 !OPAN  !(DE)             !
  21 B0!                 !LDAI  !DE               !
  22 40!                 ! ---- !40               !RESET COMMAND
  23 EE!                 !OPAN  !(DE)             !
  24 B0!                 !LDAI  !                 !
  25 4E!                 ! ---- !4E               !MODE BYTE:
  26 EE!                 !OPAN  !(DE)             !  8 DATA BITS, 1 STOP BIT,
  27 B0!                 !LDAI  !                 !  NO PARITY
  28 37!                 ! ---- !37               !COMMAND BYTE
  29 EE!                 !OPAN  !(DE)             !
  2A C7!                 !LDPI  !DE               !
  2B C2!                 ! ---- !                 !
  2C F8!                 ! ---- !FFF8             !
  2D FF!                 ! ---- !                 !
  2E EC!                 !IPAN  !(DE)             !CLEAR RECEIVER BUFFER
  2F 5A!                 !PLP   !DE               !
1030 9D!                 !PLP   !AF               !
  31 58!                 ! --   !                 !
  32 C3!                 !RTS   !                 !
  33 FF!                 !      !                 !
  34 FF!                 !      !                 !
  35 FF!                 !      !                 !
  36 FF!                 !      !                 !
  37 FF!                 !      !                 !
  38 FF!                 !      !                 !
  39 FF!                 !      !                 !
  3A FF!                 !      !                 !
  3B FF!                 !      !                 !
  3C FF!                 !      !                 !
  3D FF!                 !      !                 !
  3E FF!                 !      !                 !
  3F FF!                 !      !                 !
```

Figure 4-12. Software Routines

```
ADRdata!     LABEL      !INSTR !     MODIFIER    !           COMMENTS
-------!---------------!------!-----------------!----------------------------
1040 50!(INIT SYNC COMM)!PSP   !AF               !
  41 9C!                ! ---- !                 !
  42 52!                !PSP   !DE               !
  43 C7!                !LDPI  !DE               !INITIALIZE BAUD RATE ...
  44 C2!                ! ---- !                 !
  45 FF!                ! ---- !FFFF             !
  46 FF!                ! ---- !                 !
  47 B0!                !LDAI  !                 !
  48 36!                !      !36               !
  49 EE!                !OPAN  !(DE)             !SET TIMER 0 TO OUTPUT A SQUARE
  4A C7!                !LDPI  !DE               !  WAVE
  4B C2!                ! ---- !                 !
  4C FC!                ! ---- !FFFC             !
  4D FF!                ! ---- !                 !
  4E B0!                !LDAI  !                 !BAUD RATE=9,600
  4F 80!                ! ---- !80               !LSB OF BAUD RATE DIVISOR
1050 EE!                !OPAN  !(DE)             !
  51 B0!                !LDAI  !                 !
  52 00!                ! ---- !0                !MSB OF BAUD RATE DIVISOR
  53 EE!                !OPAN  !(DE)             !
  54 C7!                !LDPI  !DE               !RESET USART ...
  55 C2!                ! ---- !                 !
  56 F9!                ! ---- !FFF9             !
  57 FF!                ! ---- !                 !
  58 B0!                !LDAI  !                 !
  59 00!                ! ---- !0                !
  5A EE!                !OPAN  !(DE)             !
  5B B0!                !LDAI  !                 !
  5C 00!                ! ---- !0                !
  5D EE!                !OPAN  !(DE)             !
  5E B0!                !LDAI  !                 !
  5F 00!                ! ---- !0                !
1060 EE!                !OPAN  !(DE)             !
  61 B0!                !LDAI  !                 !
  62 40!                ! ---- !40               !RESET COMMAND
  63 EE!                !OPAN  !(DE)             !
  64 B0!                !LDAI  !                 !
  65 0C!                ! ---- !0C               !MODE BYTE
  66 EE!                !OPAN  !(DE)             !
  67 B0!                !LDAI  !                 !
  68 10!                ! ---- !10               !1ST SYNC CHAR (DLE)
  69 EE!                !OPAN  !(DE)             !
  6A B0!                !LDAI  !                 !
  6B 16!                ! ---- !16               !2ND SYNC CHAR (SYN)
  6C EE!                !OPAN  !(DE)             !
  6D B0!                !LDAI  !                 !
  6E B7!                ! ---- !B7               !COMMAND BYTE
  6F EE!                !OPAN  !(DE)             !
1070 B0!                !LDAI  !                 !
  71 10!                ! ---- !10               !
  72 E8!                !JSR   !(CHAR OUT)       !SEND A DLE
  73 2B!                ! ---- !                 !
  74 00!                ! ---- !                 !
  75 B0!                !LDAI  !                 !
  76 16!                ! ---- !16               !
  77 E8!                !JSR   !(CHAR OUT)       !SEND A SYN
  78 26!                ! ---- !                 !
  79 00!                ! ---- !                 !
  7A EC!SYNC LOOP       !IPAN  !(DE)             !READ STATUS
  7B 24!                !ANAI  !                 !CHECK FOR SYNC DETECT
  7C 40!                ! ---- !40               !
  7D 74!                !JPRS  !Z1               !LOOP UNTIL SYNC DETECT
  7E FB!                ! ---- !SYNC LOOP        !
  7F C7!                !LDPI  !DE
```

Figure 4-12. (Continued)

```
ADRdata!     LABEL      !INSTR !     MODIFIER    !          COMMENTS
-------!----------------!------!-----------------!----------------------------
1080 C2!                ! ---- !                 !
  81 F8!                ! ---- !FFF8              !
  82 FF!                ! ---- !                 !
  83 EC!                !IPAN  !(DE)              !CLEAR RECEIVER BUFFER
  84 5A!                !PLP   !DE                !
  85 9D!                !PLP   !AF                !
  86 58!                ! ---- !                 !
  87 C3!                !RTS   !                 !
  88 FF!                !      !                 !
  89 FF!                !      !                 !
  8A FF!                !      !                 !
  8B FF!                !      !                 !
  8C FF!                !      !                 !
  8D FF!                !      !                 !
  8E FF!                !      !                 !
  8F FF!                !      !                 !
1090 FF!                !      !                 !
  91 FF!                !      !                 !
  92 FF!                !      !                 !
  93 FF!                !      !                 !
  94 FF!                !      !                 !
  95 FF!                !      !                 !
  96 FF!                !      !                 !
  97 FF!                !      !                 !
  98 FF!                !      !                 !
  99 FF!                !      !                 !
  9A FF!                !      !                 !
  9B FF!                !      !                 !
  9C FF!                !      !                 !
  9D FF!                !      !                 !
  9E FF!                !      !                 !
  9F FF!                !      !                 !
10A0 52!(CHAR OUT)      !PSP   !DE                !
  A1 50!                !PSP   !AF                !
  A2 9C!                ! ---- !                 !
  A3 C7!                !LDPI  !DE                !
  A4 C2!                ! ---- !                 !
  A5 F9!                ! ---- !FFF9              !
  A6 FF!                ! ---- !                 !
  A7 EC!OUTPUT LOOP     !IPAN  !(DE)              !READ USART STATUS
  A8 24!                !ANAI  !                 !TRANSMITTER READY?
  A9 01!                ! ---- !1                !
  AA 74!                !JPRS  !Z1                !LOOP UNTIL READY
  AB FB!                ! ---- !OUTPUT LOOP       !
  AC C7!                !LDPI  !DE                !
  AD C2!                ! ---- !                 !
  AE F8!                ! ---- !FFF8              !
  AF FF!                ! ---- !                 !
10B0 9D!                !PLP   !AF                !
  B1 58!                ! ---- !                 !
  B2 EE!                !OPAN  !(DE)              !WRITE DATA BYTE
  B3 5A!                !PLP   !DE                !
  B4 C3!                !RTS   !                 !
  B5 FF!                !      !                 !
  B6 FF!                !      !                 !
  B7 FF!                !      !                 !
  B8 FF!                !      !                 !
  B9 FF!                !      !                 !
  BA FF!                !      !                 !
  BB FF!                !      !                 !
  BC FF!                !      !                 !
  BD FF!                !      !                 !
  BE FF!                !      !                 !
  BF FF!                !      !                 !
```

Figure 4-12. (Continued)

```
ADRdata!      LABEL       !INSTR !     MODIFIER    !           COMMENTS
-------!-----------------!------!-----------------!----------------------------
10C0 52!(CHAR IN)         !PSP   !DE               !
  C1 C7!                  !LDPI  !DE               !
  C2 C2!                  ! ---- !                 !
  C3 F9!                  ! ---- !FFF9             !
  C4 FF!                  ! ---- !                 !
  C5 EC!INPUT LOOP        !IPAN  !(DE)             !READ STATUS
  C6 24!                  !ANAI  !                 !RECEIVER READY?
  C7 02!                  ! ---- !2                !
  C8 74!                  !JPRS  !Z1               !LOOP UNTIL READY
  C9 FB!                  ! ---- !INPUT LOOP       !
  CA EC!                  !IPAN  !(DE)             !READ STATUS
  CB 24!                  !ANAI  !                 !CLEAR ZERO FLAG IF AN ERROR
  CC 38!                  ! ---- !38               !  OCCURRED
  CD C7!                  !LDPI  !DE               !
  CE C2!                  ! ---- !                 !
  CF F8!                  ! ---- !FFF8             !
10D0 FF!                  ! ---- !                 !
  D1 EC!                  !IPAN  !(DE)             !READ DATA
  D2 5A!                  !PLP   !DE               !
  D3 C3!                  !RTS   !                 !
  D4 FF!                  !      !                 !
  D5 FF!                  !      !                 !
  D6 FF!                  !      !                 !
  D7 FF!                  !      !                 !
  D8 FF!                  !      !                 !
  D9 FF!                  !      !                 !
  DA FF!                  !      !                 !
  DB FF!                  !      !                 !
  DC FF!                  !      !                 !
  DD FF!                  !      !                 !
  DE FF!                  !      !                 !
  DF FF!                  !      !                 !
10E0 50!(SYNC BLOCK OUT)  !PSP   !AF               !
  E1 9C!                  ! ---- !                 !
  E2 51!                  !PSP   !BC               !
  E3 53!                  !PSP   !HL               !
  E4 B0!                  !LDAI  !                 !
  E5 10!                  ! ---- !10               !
  E6 E8!                  !JSR   !(CHAR OUT)       !SEND DLE
  E7 B7!                  ! ---- !                 !
  E8 FF!                  ! ---- !                 !
  E9 B0!                  !LDAI  !                 !
  EA 16!                  ! ---- !16               !
  EB E8!                  !JSR   !(CHAR OUT)       !SEND SYN
  EC B2!                  ! ---- !                 !
  ED FF!                  ! ---- !                 !
  EE B0!                  !LDAI  !                 !
  EF 10!                  ! ---- !10               !
10F0 E8!                  !JSR   !(CHAR OUT)       !SEND DLE
  F1 AD!                  ! ---- !                 !
  F2 FF!                  ! ---- !                 !
  F3 B0!                  !LDAI  !                 !
  F4 02!                  ! ---- !2                !
  F5 E8!                  !JSR   !(CHAR OUT)       !SEND STX
  F6 A8!                  ! ---- !                 !
  F7 FF!                  ! ---- !                 !
  F8 C7!                  !LDPI  !HL               !
  F9 C3!                  ! ---- !                 !
  FA 00!                  ! ---- !0                !
  FB 00!                  ! ---- !                 !
  FC 89!                  !STPD  !HL               !CLEAR CRC
  FD 1E!                  ! ---- !                 !
  FE 00!                  ! ---- !CRC              !
  FF 01!                  ! ---- !                 !
```

Figure 4-12. (Continued)

```
ADRdata!     LABEL      !INSTR !     MODIFIER    !           COMMENTS
-------!----------------!------!-----------------!----------------------------
1100 5B!                !PLP   !HL               !
  01 53!                !PSP   !HL               !
  02 8A!NEXT CHAR       !LDAN  !(HL)             !GET CHARACTER
  03 07!                ! ---- !                 !
  04 43!                !ICP   !HL               !INCREMENT MEMORY REGISTER
  05 3C!                !CPAI  !                 !CHECK FOR DLE
  06 10!                ! ---- !10               !
  07 75!                !JPRS  !Z0               !JUMP IF CHAR IS NOT DLE
  08 03!                ! ---- !NOT DLE          !
  09 E8!                !JSR   !(CHAR OUT)       !SEND DLE CHARACTER
  0A 94!                ! ---- !                 !
  0B FF!                ! ---- !                 !
  0C E8!NOT DLE         !JSR   !(CHAR OUT)       !SEND CHARACTER
  0D 91!                ! ---- !                 !
  0E FF!                ! ---- !                 !
  0F E8!                !JSR   !(GEN CRC)        !UPDATE CRC
1110 AE!                ! ---- !                 !
  11 00!                ! ---- !                 !
  12 49!                !DCP   !BC               !DECREMENT COUNT
  13 8A!                !LDA   !                 !
  14 C5!                ! ---- !B                !
  15 0A!                !ORA   !                 !
  16 C1!                ! ---- !C                !
  17 75!                !JPRS  !Z0               !LOOP UNTIL COUNT=0
  18 E9!                ! ---- !NEXT CHAR        !
  19 B0!                !LDAI  !                 !
  1A 10!                ! ---- !10               !
  1B E8!                !JSR   !(CHAR OUT)       !SEND DLE
  1C 82!                ! ---- !                 !
  1D FF!                ! ---- !                 !
  1E B0!                !LDAI  !                 !
  1F 03!                ! ---- !3                !
1120 E8!                !JSR   !(CHAR OUT)       !SEND ETX
  21 7D!                ! ---- !                 !
  22 FF!                ! ---- !                 !
  23 E8!                !JSR   !(GEN CRC)        !UPDATE CRC
  24 9A!                ! ---- !                 !
  25 00!                ! ---- !                 !
  26 A0!                !LDAD  !                 !GET LSB OF CRC
  27 00!                ! ---- !CRC              !
  28 01!                ! ---- !                 !
  29 E8!                !JSR   !(CHAR OUT)       !SEND LSB OF CRC
  2A 74!                ! ---- !                 !
  2B FF!                ! ---- !                 !
  2C A0!                !LDAD  !                 !GET MSB OF CRC
  2D 01!                ! ---- !CRC+1            !
  2E 01!                ! ---- !                 !
  2F E8!                !JSR   !(CHAR OUT)       !SEND MSB OF CRC
1130 6E!                ! ---- !                 !
  31 FF!                ! ---- !                 !
  32 5B!                !PLP   !HL               !
  33 59!                !PLP   !BC               !
  34 9D!                !PLP   !AF               !
  35 58!                ! ---- !                 !
  36 C3!                !RTS   !                 !
  37 FF!                !      !                 !
  38 FF!                !      !                 !
  39 FF!                !      !                 !
  3A FF!                !      !                 !
  3B FF!                !      !                 !
  3C FF!                !      !                 !
  3D FF!                !      !                 !
  3E FF!                !      !                 !
  3F FF!                !      !                 !
```

Figure 4-12. (Continued)

```
ADRdata!     LABEL       !INSTR !     MODIFIER    !          COMMENTS
-------!----------------!------!-----------------!----------------------------
1140 52!(SYNC BLOCK IN) !PSP   !DE               !
  41 53!                !PSP   !HL               !
  42 C7!                !LDPI  !DE               !
  43 C2!                ! ---- !                 !
  44 F9!                ! ---- !FFF9             !
  45 FF!                ! ---- !                 !
  46 B0!                !LDAI  !                 !USART COMMAND BYTE TO RESET
  47 37!                ! ---- !37               !  ERROR FLAGS
  48 EE!                !OPAN  !(DE)             !RESET USART ERROR FLAGS
  49 C7!                !LDPI  !BC               !CLEAR COUNT
  4A C1!                ! ---- !                 !
  4B 00!                ! ---- !0                !
  4C 00!                ! ---- !                 !
  4D E8!GET DLE         !JSR   !(CHAR IN)        !GET CHARACTER
  4E 70!                ! ---- !                 !
  4F FF!                ! ---- !                 !
1150 3C!                !CPAI  !                 !IS IT A DLE?
  51 10!                ! ---- !10               !
  52 75!                !JPRS  !Z0               !LOOP UNTIL A DLE IS RECEIVED
  53 F9!                ! ---- !GET DLE          !
  54 E8!                !JSR   !(CHAR IN)        !GET CHARACTER
  55 69!                ! ---- !                 !
  56 FF!                ! ---- !                 !
  57 3C!                !CPAI  !                 !IS IT AN STX?
  58 02!                ! ---- !2                !
  59 75!                !JPRS  !Z0               !LOOP UNTIL DLE.STX ARE
  5A F2!                ! ---- !GET DLE          !  RECEIVED
  5B B0!                !LDAI  !                 !
  5C 00!                ! ---- !0                !
  5D A2!                !STAD  !                 !
  5E 00!                ! ---- !CRC              !
  5F 01!                ! ---- !                 !
1160 A2!                !STAD  !                 !
  61 01!                ! ---- !CRC+1            !
  62 01!                ! ---- !                 !
  63 E8!NEXT            !JSR   !(CHAR IN)        !GET A CHARACTER
  64 5A!                ! ---- !                 !
  65 FF!                ! ---- !                 !
  66 3C!                !CPAI  !                 !CHECK FOR DLE
  67 10!                ! ---- !10               !
  68 75!                !JPRS  !Z0               !JUMP IF NOT DLE
  69 0F!                ! ---- !PROC CHAR        !
  6A E8!                !JSR   !(CHAR IN)        !GET SECOND CHAR. OF DLE
  6B 53!                ! ---- !                 !  SEQUENCE
  6C FF!                ! ---- !                 !
  6D 3C!                !CPAI  !                 !CHECK FOR SYN
  6E 16!                ! ---- !16               !
  6F 74!                !JPRS  !Z1               !IGNORE DLE.SYN AND GET NEXT
1170 F2!                ! ---- !NEXT             !  CHAR
  71 3C!                !CPAI  !                 !CHECK FOR ETX
  72 03!                ! ---- !3                !
  73 74!                !JPRS  !Z1               !END OF TRANSMISIION IF ETX
  74 0D!                ! ---- !DONE             !  RECEIVED
  75 3C!                !CPAI  !                 !CHECK FOR DLE
  76 10!                ! ---- !10               !
  77 75!                !JPRS  !Z0               !ERROR OCCURRED IF NONE OF THE
  78 2B!                ! ---- !ERROR            !  ABOVE WAS RECEIVED
  79 E8!PROC CHAR       !JSR   !(GEN CRC)        !UPDATE CRC
  7A 44!                ! ---- !                 !
  7B 00!                ! ---- !                 !
  7C 88!                !STAN  !(HL)             !STORE CHARACTER
  7D 07!                ! ---- !                 !
  7E 43!                !ICP   !HL               !INCREMENT MEMORY POINTER
  7F 41!                !ICP   !BC               !INCREMENT COUNTER
```

Figure 4-12. (Continued)

```
ADRdata!     LABEL      !INSTR !     MODIFIER    !           COMMENTS
-------!----------------!------!-----------------!-----------------------------
1180 EB!                !JPRS  !UN               !GET THE NEXT CHARACTER
  81 E1!                !      !NEXT             !
  82 E8!DONE            !JSR   !(GEN CRC)        !UPDATE CRC FOR ETX CHARACTER
  83 3B!                ! ---- !                 !
  84 00!                ! ---- !                 !
  85 E8!                !JSR   !(CHAR IN)        !GET LSB OF CRC
  86 38!                ! ---- !                 !
  87 FF!                ! ---- !                 !
  88 E8!                !JSR   !(GEN CRC)        !UPDATE CRC
  89 35!                ! ---- !                 !
  8A 00!                ! ---- !                 !
  8B E8!                !JSR   !(CHAR IN)        !GET MSB OF CRC
  8C 32!                ! ---- !                 !
  8D FF!                ! ---- !                 !
  8E E8!                !JSR   !(GEN CRC)        !UPDATE CRC
  8F 2F!                ! ---- !                 !
1190 00!                ! ---- !                 !
  91 A0!                !LDAD  !                 !GET CUMULATIVE CRC
  92 00!                ! ---- !CRC              !
  93 01!                ! ---- !                 !
  94 8A!                !LDR   !D,A              !
  95 F0!                ! ---- !                 !
  96 A0!                !LDAD  !                 !
  97 01!                ! ---- !CRC+1            !
  98 01!                ! ---- !                 !
  99 0A!                !ORA   !                 !
  9A C6!                ! ---- !D                !
  9B 75!                !JPRS  !Z0               !JUMP TO ERROR IF CRC NOT 0
  9C 07!                ! ---- !ERROR            !
  9D C7!                !LDPI  !DE               !
  9E C2!                ! ---- !                 !
  9F F9!                ! ---- !FFF9             !
11A0 FF!                ! ---- !                 !
  A1 EC!                !IPAN  !(DE)             !READ USART STATUS
  A2 24!                !ANAI  !                 !TEST ERROR FLAGS
  A3 38!                ! ---- !38               !
  A4 5B!ERROR           !PLP   !HL               !RETURN WITH ZERO FLAG RESET IF
  A5 5A!                !PLP   !DE               !  AN ERROR OCCURRED
  A6 C3!                !RTS   !                 !
  A7 FF!                !      !                 !
  A8 FF!                !      !                 !
  A9 FF!                !      !                 !
  AA FF!                !      !                 !
  AB FF!                !      !                 !
  AC FF!                !      !                 !
  AD FF!                !      !                 !
  AE FF!                !      !                 !
  AF FF!                !      !                 !
11B0 FF!                !      !                 !
  B1 FF!                !      !                 !
  B2 FF!                !      !                 !
  B3 FF!                !      !                 !
  B4 FF!                !      !                 !
  B5 FF!                !      !                 !
  B6 FF!                !      !                 !
  B7 FF!                !      !                 !
  B8 FF!                !      !                 !
  B9 FF!                !      !                 !
  BA FF!                !      !                 !
  BB FF!                !      !                 !
  BC FF!                !      !                 !
  BD FF!                !      !                 !
  BE FF!                !      !                 !
  BF FF!                !      !                 !
```

Figure 4-12. (Continued)

```
ADRdata!     LABEL      !INSTR !     MODIFIER    !          COMMENTS
-------!----------------!------!-----------------!---------------------------
11C0 50!(GEN CRC)       !PSP   !AF               !
  C1 9C!                ! ---- !                 !
  C2 51!                !PSP   !BC               !
  C3 53!                !PSP   !HL               !
  C4 B1!                !LDRI  !C                !SET COUNT=8
  C5 08!                ! ---- !8                !
  C6 8B!                !LDPD  !HL               !LOAD CUMULATIVE CRC
  C7 1E!                ! ---- !                 !
  C8 00!                ! ---- !CRC              !
  C9 01!                ! ---- !                 !
  CA D0!GEN BIT         !RRA   !                 !ROTATE DATA RIGHT
  CB C8!                ! ---- !                 !
  CC 73!                !JPRS  !C0               !JUMP IF CARRY = 0
  CD 0A!                ! ---- !GC1              !
  CE 8A!                !LDR   !B,A              !SAVE DATA BYTE
  CF E8!                !      !                 !
11D0 8A!                !LDA   !                 !CRC=CRC XOR 0001H
  D1 C3!                ! ---- !L                !
  D2 34!                !XRAI  !                 !
  D3 01!                ! ---- !1                !
  D4 8A!                !LDR   !L,A              !
  D5 D8!                ! ---- !                 !
  D6 8A!                !LDA   !                 !RESTORE DATA BYTE
  D7 C5!                ! ---- !B                !
  D8 8A!GC1             !LDR   !B,A              !SAVE DATA BYTE
  D9 E8!                ! ---- !                 !
  DA 8A!                !LDA   !                 !
  DB C3!                ! ---- !L                !
  DC D0!                !RRAC  !                 !ROTATE CRC RIGHT
  DD D8!                ! ---- !                 !
  DE 8A!                !LDR   !L,A              !
  DF D8!                ! ---- !                 !
11E0 8A!                !LDA   !                 !
  E1 C7!                ! ---- !H                !
  E2 D0!                !RRAC  !                 !
  E3 D8!                ! ---- !                 !
  E4 8A!                !LDR   !H,A              !
  E5 F8!                ! ---- !                 !
  E6 8A!                !LDA   !                 !
  E7 C3!                ! ---- !L                !
  E8 73!                !JPRS  !C0               !
  E9 04!                ! ---- !GC1A             !
  EA 0C!                !ORAI  !                 !
  EB 80!                ! ---- !80               !
  EC EB!                !JPRS  !UN               !
  ED 02!                ! ---- !GC1B             !
  EE 24!GC1A            !ANAI  !                 !
  EF 7F!                ! ---- !7F               !
11F0 8A!GC1B            !LDR   !L,A              !
  F1 D8!                ! ---- !                 !
  F2 8A!                !LDA   !                 !
  F3 C7!                ! ---- !H                !
  F4 D0!                !RLA   !                 !
  F5 C0!                ! ---- !                 !
  F6 8A!                !LDA   !                 !RESTORE DATA BYTE
  F7 C5!                ! ---- !B                !
  F8 73!                !JPRS  !C0               !JUMP IF CARRY=0
  F9 10!                ! ---- !GC2              !
  FA 8A!                !LDR   !B,A              !SAVE DATA BYTE
  FB E8!                ! ---- !                 !
  FC 8A!                !LDA   !                 !CRC=CRC XOR 2001H
  FD C7!                ! ---- !H                !
  FE 34!                !XRAI  !                 !
  FF 20!                ! ---- !20               !
```

Figure 4-12. (Continued)

```
ADRdata!     LABEL      !INSTR !     MODIFIER   !           COMMENTS
-------!----------------!------!----------------!-----------------------------
1200 8A!                !LDR   !H,A             !
  01 F8!                ! ---- !               !
  02 8A!                !LDA   !               !
  03 C3!                ! ---- !L              !
  04 34!                !XRAI  !               !
  05 01!                ! ---- !1              !
  06 8A!                !LDR   !L,A            !
  07 D8!                ! ---- !               !
  08 8A!                !LDA   !B              !RESTORE DATA BYTE
  09 C5!                ! ---- !               !
  0A FE!GC2             !DCR   !C              !DECREMENT COUNT
  0B C9!                ! ---- !               !
  0C 75!                !JPRS  !Z0             !LOOP UNTIL COUNT=0
  0D BC!                ! ---- !GEN BIT        !
  0E 89!                !STPD  !HL             !STORE CRC
  0F 1E!                ! ---- !               !
1210 00!                ! ---- !CRC            !
  11 01!                ! ---- !               !
  12 5B!                !PLP   !HL             !
  13 59!                !PLP   !BC             !
  14 9D!                !PLP   !AF             !
  15 58!                ! ---- !               !
  16 C3!                !RTS   !               !
  17 FF!                !      !               !
  18 FF!                !      !               !
  19 FF!                !      !               !
  1A FF!                !      !               !
  1B FF!                !      !               !
  1C FF!                !      !               !
  1D FF!                !      !               !
  1E FF!                !      !               !
  1F FF!                !      !               !
1220 50!(DELAY)         !PSP   !AF             !
  21 9C!                ! ---- !               !
  22 52!                !PSP   !DE             !
  23 C7!                !LDPI  !DE             !SET UP TIMER 1
  24 C2!                ! ---- !               !
  25 FF!                ! ---- !FFFF           !
  26 FF!                ! ---- !               !
  27 B0!                !LDAI  !               !
  28 70!                ! ---- !70             !
  29 EE!                !OPAN  !(DE)           !
  2A C7!                !LDPI  !DE             !
  2B C2!                ! ---- !               !
  2C FD!                ! ---- !FFFD           !
  2D FF!                ! ---- !               !
  2E 8A!                !LDA   !               !
  2F C1!                ! ---- !C              !GET LSB OF DELAY COUNT
1230 EE!                !OPAN  !(DE)           !SEND LSB TO TIMER
  31 8A!                !LDA   !               !
  32 C5!                ! ---- !B              !GET MSB OF DELAY COUNT
  33 EE!                !OPAN  !(DE)           !SEND MSB TO TIMER
  34 5A!                !PLP   !DE             !
  35 9D!                !PLP   !AF             !
  36 58!                ! ---- !               !
  37 9B!                !HLTT  !               !WAIT UNTIL TIMER TIMES OUT
  38 C3!                !RTS   !               !
  39 FF!                !      !               !
  3A FF!                !      !               !
  3B FF!                !      !               !
  3C FF!                !      !               !
  3D FF!                !      !               !
  3E FF!                !      !               !
  3F FF!                !      !               !
```

Figure 4-12. (Continued)

```
ADRdata!     LABEL       !INSTR !     MODIFIER    !          COMMENTS
-------!----------------!------!-----------------!----------------------------
1240 C7!(INIT INTR)      !LDPI  !HL               !
  41 C3!                 ! ---- !                 !
  42 80!                 ! ---- !INTR OFFSET      !
  43 12!                 ! ---- !                 !
  44 89!                 !STPD  !HL               !STORE OFFSET OF INTERRUPT
  45 1E!                 ! ---- !                 !  AT INTERRUPT VECTOR LOCATION
  46 14!                 ! ---- !INT5             !
  47 00!                 ! ---- !                 !
  48 C7!                 !LDPI  !HL               !
  49 C3!                 ! ---- !                 !
  4A 00!                 ! ---- !INTR SEGMENT     !
  4B 00!                 ! ---- !                 !
  4C 89!                 !STPD  !HL               !STORE SEGMENT OF INTERRUPT
  4D 1E!                 ! ---- !                 !  ROUTINE AT INTERRUPT VECTOR
  4E 16!                 ! ---- !INT5+2           !  LOCATION
  4F 00!                 ! ---- !                 !
1250 C7!                 !LDPI  !DE               !
  51 C2!                 ! ---- !                 !
  52 FF!                 ! ---- !FFFF             !
  53 FF!                 ! ---- !                 !
  54 B0!                 !LDAI  !                 !
  55 B0!                 ! ---- !B0               !
  56 EE!                 !OPAN  !(DE)             !SET TIMER FOR INTERRUPTS
  57 C7!                 !LDPI  !DE               !
  58 C2!                 ! ---- !                 !
  59 FE!                 ! ---- !FFFE             !
  5A FF!                 ! ---- !                 !
  5B B0!                 !LDAI  !                 !LOAD DELAY COUNT:
  5C XX!                 ! ---- !DELAY LO         !
  5D EE!                 !OPAN  !(DE)             !DELAY=(DELAY LO+256* DELAY HI)
  5E B0!                 !LDAI  !                 !  *813.8 NS
  5F XX!                 ! ---- !DELAY HI         !
1260 EE!                 !OPAN  !(DE)             !
  61 FB!                 !ENI   !                 !ENABLE INTERRUPTS
  62 C3!                 !RTS   !                 !
  63 FF!                 !      !                 !
  64 FF!                 !      !                 !
  65 FF!                 !      !                 !
  66 FF!                 !      !                 !
  67 FF!                 !      !                 !
  68 FF!                 !      !                 !
  69 FF!                 !      !                 !
  6A FF!                 !      !                 !
  6B FF!                 !      !                 !
  6C FF!                 !      !                 !
  6D FF!                 !      !                 !
  6E FF!                 !      !                 !
  6F FF!                 !      !                 !
1270 FF!                 !      !                 !
  71 FF!                 !      !                 !
  72 FF!                 !      !                 !
  73 FF!                 !      !                 !
  74 FF!                 !      !                 !
  75 FF!                 !      !                 !
  76 FF!                 !      !                 !
  77 FF!                 !      !                 !
  78 FF!                 !      !                 !
  79 FF!                 !      !                 !
  7A FF!                 !      !                 !
  7B FF!                 !      !                 !
  7C FF!                 !      !                 !
  7D FF!                 !      !                 !
  7E FF!                 !      !                 !
  7F FF!                 !      !                 !
```

Figure 4-12. (Continued)

```
ADRdata!     LABEL      !INSTR !     MODIFIER    !           COMMENTS
-------!----------------!------!-----------------!-----------------------------
1280 52!(INTR)          !PSP   !DE               !
  81 50!                !PSP   !AF               !
  82 9C!                ! ---- !                 !
  83 C7!                !LDPI  !DE               !POINT TO TIMER 2 COUNTER
  84 C2!                ! ---- !                 !
  85 FE!                ! ---- !FFFE             !
  86 FF!                ! ---- !                 !
  87 B0!                !LDA   !                 !LOAD DELAY COUNT:
  88 XX!                ! ---- !DELAY LO         !
  89 EE!                !OPAN  !(DE)             !DELAY=(DELAY LO+256* DELAY HI)
  8A B0!                !LDA   !                 !  *813.8 NS
  8B XX!                ! ---- !DELAY HI         !
  8C EE!                !OPAN  !(DE)             !
  8D 90!                !NOP   !                 !
  8E 90!                !NOP   !                 !
  8F 90!                !NOP   !                 !
1290 90!                !NOP   !                 !
  91 90!                !NOP   !                 !
  92 90!                !NOP   !                 !
  93 90!                !NOP   !                 !PLACE INTERRUPT ROUTINE HERE
  94 90!                !NOP   !                 !
  95 90!                !NOP   !                 !
  96 90!                !NOP   !                 !
  97 90!                !NOP   !                 !
  98 90!                !NOP   !                 !
  99 90!                !NOP   !                 !
  9A 90!                !NOP   !                 !
  9B 90!                !NOP   !                 !
  9C 9D!                !PLP   !AF               !
  9D 58!                ! ---- !                 !
  9E 5A!                !PLP   !DE               !
  9F CF!                !RTI   !                 !
12A0 FF!                !      !                 !
  A1 FF!                !      !                 !
  A2 FF!                !      !                 !
  A3 FF!                !      !                 !
  A4 FF!                !      !                 !
  A5 FF!                !      !                 !
  A6 FF!                !      !                 !
  A7 FF!                !      !                 !
  A8 FF!                !      !                 !
  A9 FF!                !      !                 !
  AA FF!                !      !                 !
  AB FF!                !      !                 !
  AC FF!                !      !                 !
  AD FF!                !      !                 !
  AE FF!                !      !                 !
  AF FF!                !      !                 !
12B0 FF!                !      !                 !
  B1 FF!                !      !                 !
  B2 FF!                !      !                 !
  B3 FF!                !      !                 !
  B4 FF!                !      !                 !
  B5 FF!                !      !                 !
  B6 FF!                !      !                 !
  B7 FF!                !      !                 !
  B8 FF!                !      !                 !
  B9 FF!                !      !                 !
  BA FF!                !      !                 !
  BB FF!                !      !                 !
  BC FF!                !      !                 !
  BD FF!                !      !                 !
  BE FF!                !      !                 !
  BF FF!                !      !                 !
```

Figure 4-12. (Continued)

REFERENCE DRAWINGS

The schematic (figure 5-1) and assembly drawing (figure 5-2) in the following pages are included in this manual FOR REFERENCE USE ONLY. These drawings may differ in some respects from the card you receive from Pro-Log. If you require the specific revision documents of the schematic and assembly for the board shipped, please use the information request form in the back of this manual to order them.

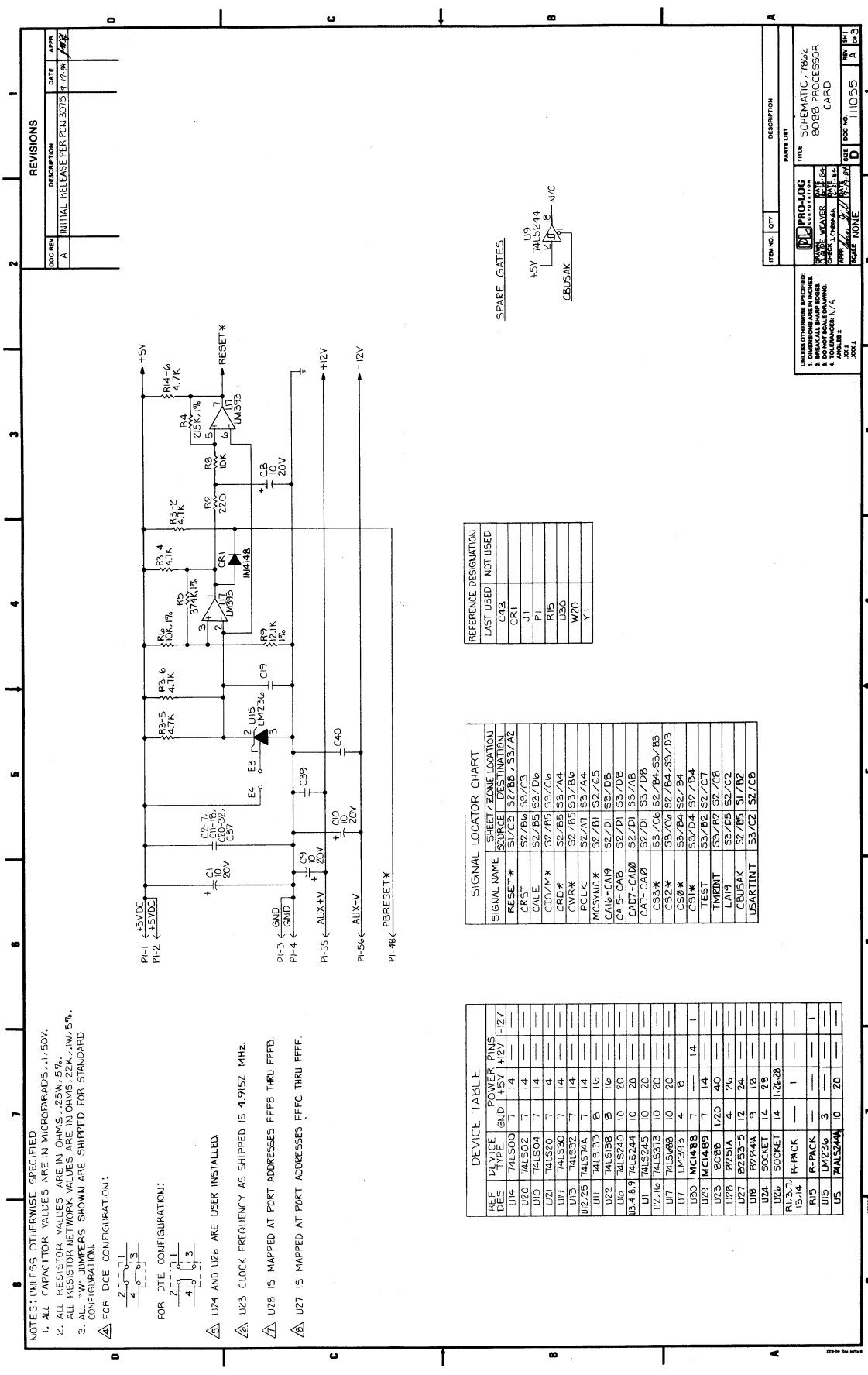


Figure 5-1. 7862 Schematic, Page 1 of 3
FOR REFERENCE ONLY

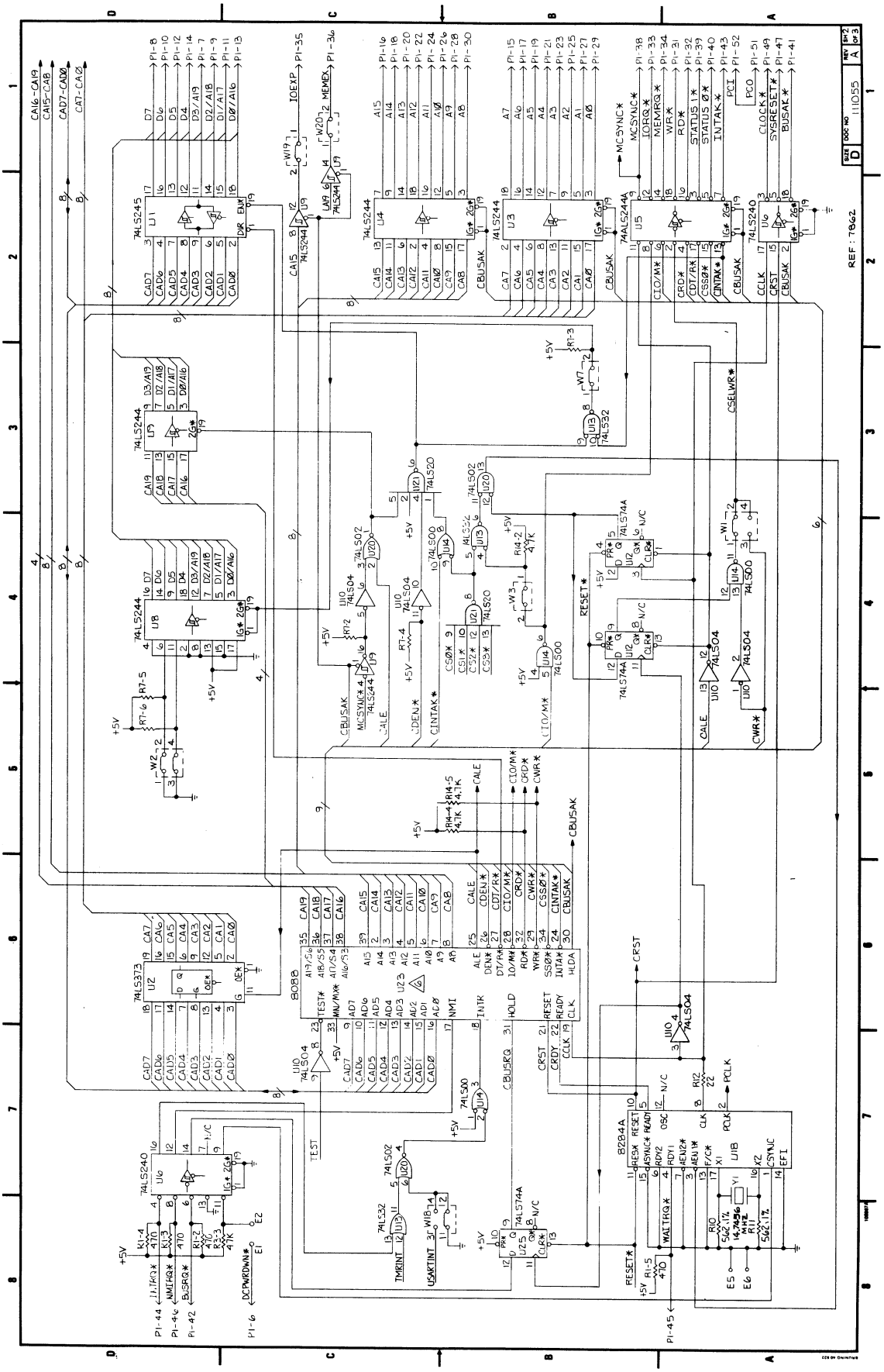


Figure 5-1. 7862 Schematic, Page 2 of 3

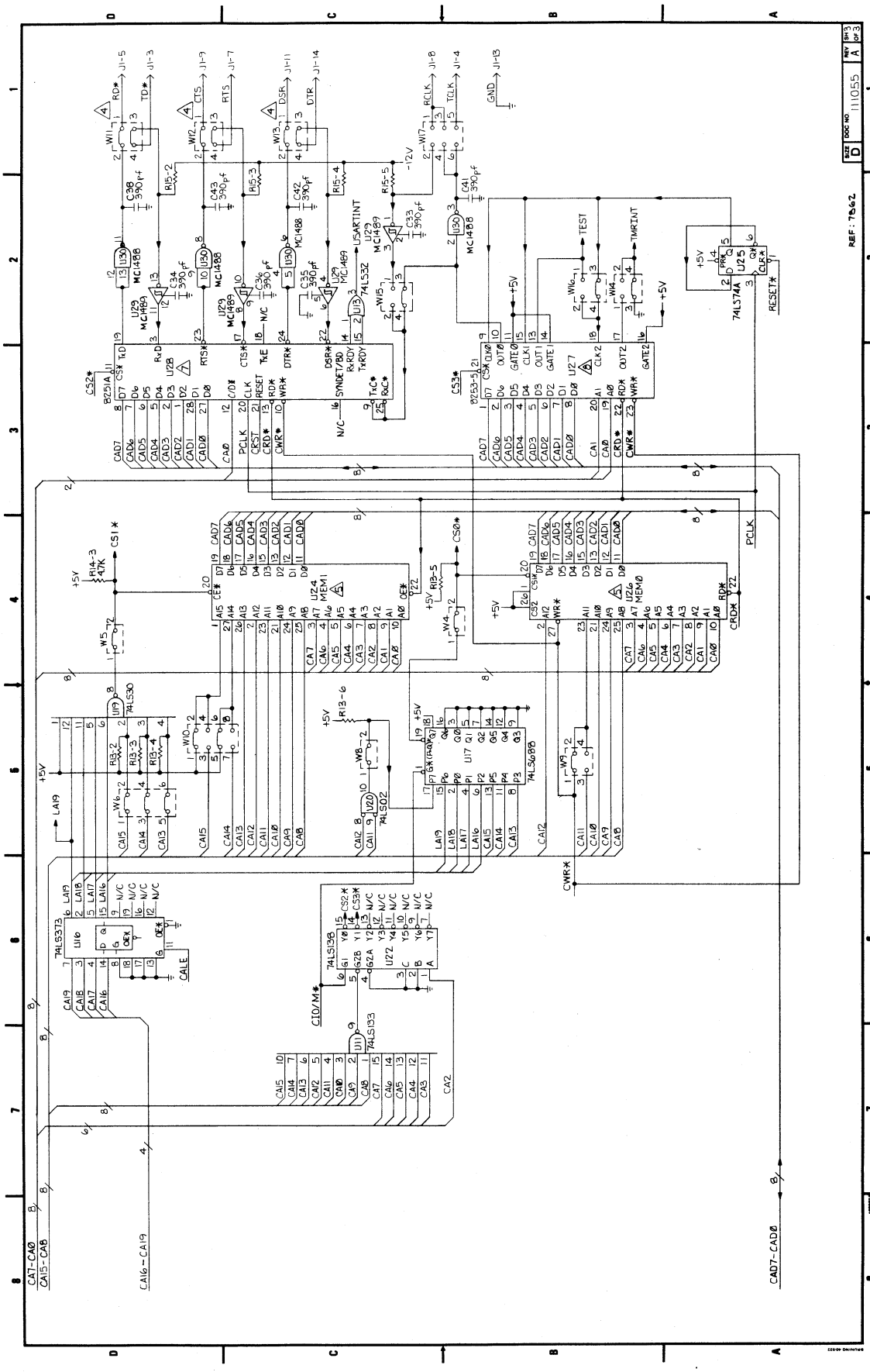
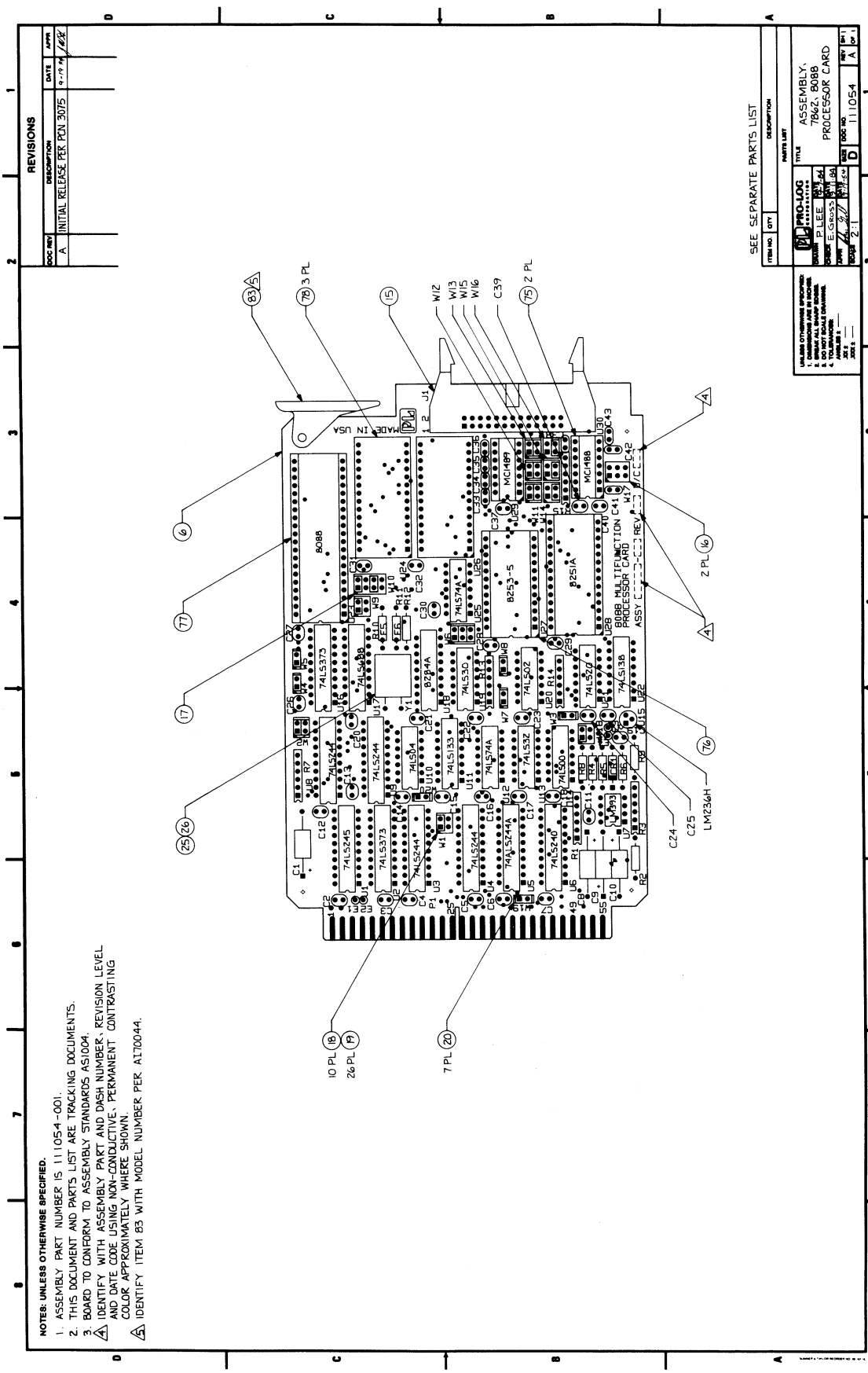


Figure 5-1. 7862 Schematic, Page 3 of 3


REVISIONS
DOC REV
DESCRIPTION
DATE
APPR
A
INITIAL RELEASE PER PCN 3075
SEE SEPARATE PARTS LIST
PRO-LOG
ASSEMBLY, 7862, 8088 PROCESSOR CARD
DOC NO. 111054
REV A
SIZE D
SCALE 2:1
NOTES: UNLESS OTHERWISE SPECIFIED.
1. ASSEMBLY PART NUMBER IS 111054-001.
2. THIS DOCUMENT AND PARTS LIST ARE TRACKING DOCUMENTS.
3. BOARD TO CONFORM TO ASSEMBLY STANDARDS AS1004.
4. IDENTIFY WITH ASSEMBLY PART AND DASH NUMBER, REVISION LEVEL AND DATE CODE USING NON-CONDUCTIVE, PERMANENT CONTRASTING COLOR APPROXIMATELY WHERE SHOWN.
5. IDENTIFY ITEM 83 WITH MODEL NUMBER PER A170044.
8088
8284A
8253-5
8251A
MC1489
MC1488
74LS373
74LS244
74LS245
74LS374A
74LS138
74LS32
74LS02
74LS04
74LS30
74LS240
74LS133
74LS08
74LS00
74LS74A
74LS393
74LS922
74LS244A
74LS374
74LS30
74LS393
74LS922
74LS244A
74LS374A
74LS138
74LS32
74LS02
74LS04
74LS30
74LS240
74LS133
74LS08
74LS00
74LS74A

RETURN FOR REPAIR PROCEDURES

Domestic Customers:

1. Call CUSTOMER SERVICE at (408) 372-4593 (in California) or (800) 538-9570 (outside of California) 7 AM to 3 PM PST.

2. Explain the problem and we may be able to solve it on the phone. If not, we will give you a Customer Return Order (CRO) number.

   Mark the CRO number on the shipping label, packing slip, and other paperwork accompanying the return. We cannot accept returns without a CRO.

3. Please be sure to enclose a packing slip with the CRO number, the equipment serial number if applicable, and the reason for return. Also enclose the name and telephone number of the person we should contact (preferably the user).

4. Package the equipment in a solid cardboard box secured with packing material.

CAUTION: Loose MOS integrated circuits, or any product containing CMOS integrated circuits, must be protected from electrostatic discharge during handling. Use conductive foam pads or conductive plastic bags, and never place MOS or CMOS circuitry in contact with Styrofoam materials.

5. Ship prepaid and insured to:

   Pro-Log Corporation
   2411 Garden Road
   Monterey, CA 93940
   Reference CRO #_____.

International Customers:

Equipment repair is handled by your local Pro-Log distributor. If you need to contact Pro-Log, the factory can be reached at any time by TWX at 910-360-7082.

LIMITED WARRANTY:

Seller warrants that the articles furnished hereunder are free from defects in material and workmanship and perform to applicable, published Pro-Log specifications for two years from date of shipment. This warranty is in lieu of any other warranty expressed or implied. In no event will Seller be liable for special or consequential damages as a result of any alleged breach of this warranty provision. This liability of Seller hereunder shall be limited to replacing or repairing, at its option, any defective units which are returned F.O.B. Seller's plant. Equipment or parts which have been subject to abuse, misuse, accident, alteration, neglect, unauthorized repair, or installation are not covered by warranty. Seller shall have the right of final determination as to the existence and cause of defect. As to items repaired or replaced, the warranty shall continue in effect for the remainder of the warranty period, or for ninety (90) days following date of shipment by Seller of the repaired or replaced part whichever period is longer. No liability is assumed for expendable items such as lamps and fuses. No warranty is made with respect to custom equipment or products produced to buyer's specifications except as specifically stated in writing by Seller and contained in the contract.

INTRODUCTION

Appendix A is an overview of the 8088 microprocessor. It covers CPU architecture and operation including:

- Instruction Pipeline
- Registers
- Reset
- Segmentation
- Memory Addressing
- I/O Structure
- Interrupts
- System Data Flow

For more information on the architecture and operation of the 8088 microprocessor, refer to the Intel iAPX88 with an Introduction to the 188, Reston Publishing Company, Inc., Reston, Virginia. #R3016-4.

ARCHITECTURE

Instruction Pipeline

Most 8-bit microprocessors (8085, Z80, and 6809) fetch and execute instructions in a sequential manner. The fetching CPU hardware must wait for the execution to finish before it can fetch the next instruction. The executing CPU hardware must wait for an instruction to be fetched before it can execute it. The 8088 eliminates waiting time by dividing the internal CPU into two independent functional blocks. This is shown in figure A-1.

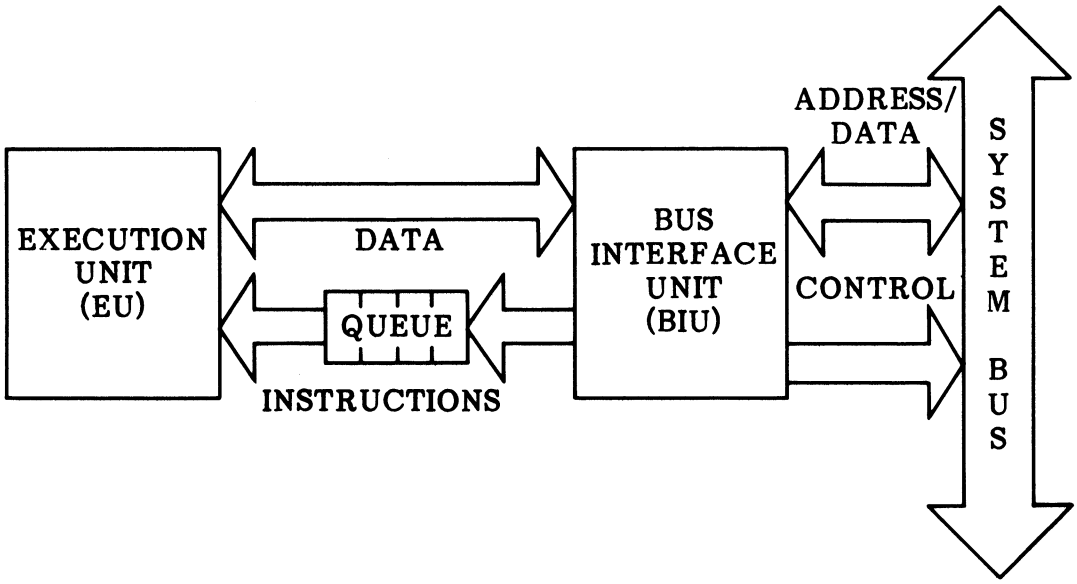


Figure A-1. Pipeline Structure

The CPU contains an Execution Unit (EU), a Bus Interface Unit (BIU), and a buffer (queue) between EU and BIU.

The EU has no connection to the system bus. It gets instructions from the queue, which is maintained by the BIU. Likewise, when an instruction requires access to memory or I/O devices, the EU requests the BIU to obtain or store the data.

The BIU performs all bus operations for the EU. Data is transferred between the CPU and memory or I/O devices upon demand from the EU. In addition, during periods when the EU is busy executing instructions, the BIU pre-fetches the next sequential instructions from memory.

The queue is a 4-byte internal RAM array in which the BIU stores the pre-fetched instructions. The BIU fetches an instruction byte whenever one byte of the queue is empty and there is no active request for bus access from the EU.

Under most circumstances, the queue contains at least one byte of an instruction so the EU does not have to wait for instructions to be fetched. The instructions placed in queue are sequentially fetched from memory — that is, they are the next logical instruction as long as execution proceeds serially. If the EU executes an instruction that transfers control to another location, the BIU empties the queue, fetches the instruction from the new address, passes it directly to the EU, and begins refilling the queue from the new location.

Registers

The 8088 register set consists of fourteen 16-bit registers. These registers can be divided into three functional groups as seen in figure A-2. The functional groups are General Registers, Control Registers, and Segment Registers.

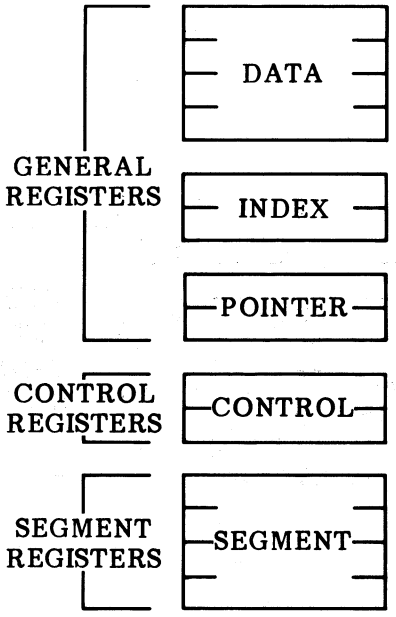


Figure A-2. 8088 Register Set

General Registers: The general register set consists of eight 16-bit registers. As seen in figure A-3, the general register set includes data registers, pointer registers, and index registers.

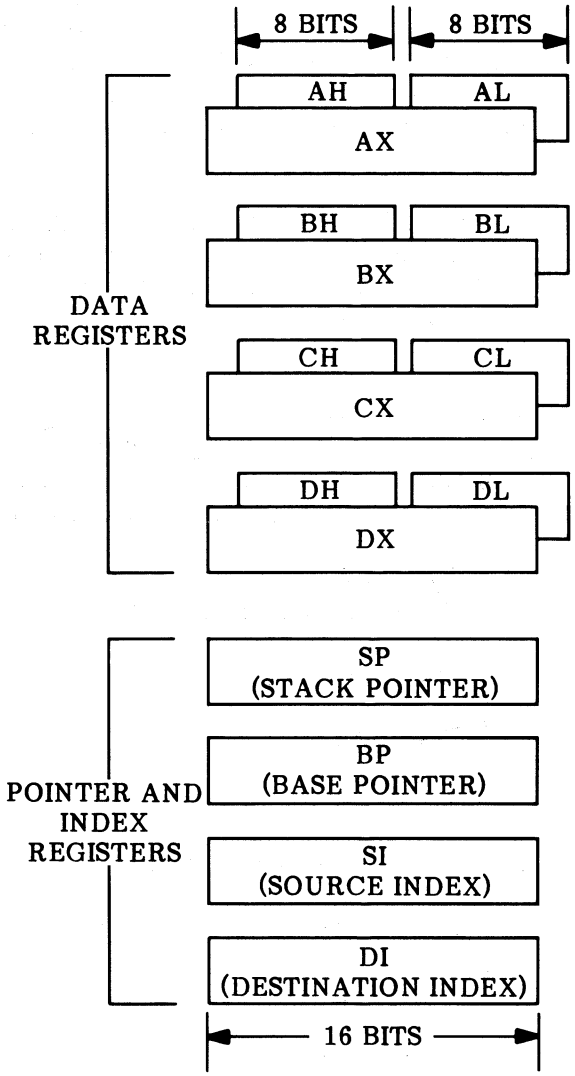


Figure A-3. General Registers

The data registers are unique in that each can be addressed as a 16-bit register or two 8-bit registers. The 16-bit registers are AX, BX, CX, and DX. The 8-bit registers are AH, AL, BH, BL, CH, CL, DH, and DL. The suffix "H" represents the most significant (high) eight bits of the 16-bit register and the suffix "L" represents the least significant (low) byte.

The data registers can be used for most arithmetic and logic operations. Some of the 8088 instructions make implicit use of the data registers. For example, the CX register is frequently used to contain a count value during repetitive instructions. Figure A-4 lists the implicit use of the general registers.

The pointer and index registers consist of the 16-bit registers SP, BP, SI, and DI, as shown in figure A-3. These registers can also be used in most arithmetic and logic operations, although they usually contain an offset address for addressing within a segment.

Most microprocessors have a single stack pointer register called the SP. The 8088 has an SP register and an additional pointer into the stack called the BP or base pointer register. The BP register can be used to reference the stack independently from the SP register. Using the separate BP register to mark the stack is a convenient way to pass parameters on the stack between a calling routine and a subroutine.

There are two index registers called the source index (SI) and destination index (DI). These 16-bit registers are used implicitly by string instructions. They also build some of the more powerful addressing modes. Both SI and DI have auto-incrementing and auto-decrementing capabilities.

REGISTER	OPERATIONS
AX	Word Multiply, Word Divide, Word I/O
AL	Byte Multiply, Byte Divide, Byte I/O, Translate, Decimal Arithmetic
AH	Byte Multiply, Byte Divide
BX	Translate
CX	String Operations, Loops
CL	Variable Shift and Rotate
DX	Word Multiply, Word Divide, Indirect I/O
SP	Stack Operations
SI	String Operations
DI	String Operations

Figure A-4. Implicit Use of General Registers

Control Registers: The instruction pointer and flag register are each 16 bits, as shown in figure A-5. The instruction pointer (IP) is similar to a program counter (PC) in other microprocessors except the IP points to the next instruction to be fetched, whereas the PC points to the next instruction to be executed. This difference is due to the pre-fetching of instructions into the internal queue. Whenever the IP is saved on the stack, it is first automatically adjusted to point to the next instruction to be executed.

The flag register contains six status flags and three control flags. The six status flags are updated by the EU to reflect certain properties of the result of an arithmetic or logic operation. The flags are defined as follows:

C (Bit 0) — The carry flag is set when an arithmetic instruction causes a carry out of or a borrow into the most significant bit of the result. The carry flag may also be used to isolate a bit by using rotate instructions.

P (Bit 2) — The parity flag is set when the result of an operation has even parity (an even number of bits set to "one").

A (Bit 4) — The auxiliary carry flag is set when an operation results in a carry out of or a borrow into the lower nibble of the result.

Z (Bit 6) — The zero flag is set when the result of an operation is zero.

S (Bit 7) — The sign flag is set when the result of an operation has a high order bit set to "one."

O (Bit 11) — The overflow bit is set when an arithmetic overflow has occurred. An arithmetic overflow is defined as the loss of a significant digit because the result was too large for the destination.

The three remaining bits are control bits. They may be set and cleared by software to control the processor operation:

T (Bit 8) — Setting the trap flag places the processor in a single-step mode. In the single-step mode, the processor generates an internal interrupt after each instruction. An interrupt service routine can then be written to view the effect of the instruction.

I (Bit 9) — Setting the interrupt flag allows the processor to recognize external interrupt requests. Clearing the interrupt flags masks the external interrupt requests.

D (Bit 10) — Setting the direction flag causes the SI and DI registers to auto-decrement during string instructions. Clearing the direction flag causes the SI and DI registers to auto-increment during string instructions.

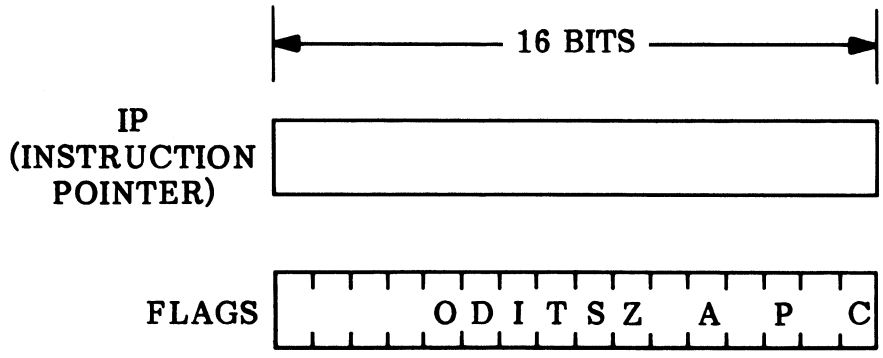


Figure A-5. Control Registers

Segment Registers: The 8088 provides 20 address lines to address up to one megabyte of memory. The megabyte is divided into logical segments of up to 64K bytes each. The CPU has direct access to four segments at a time. The starting address (the upper 16 bits of the 20-bit address) for each of the four segments is contained within the segment registers. The four segments are the code segment (CS), data segment (DS), stack segment (SS), and extra segment (ES). Refer to Segmentation, page A-7, for more information on how the segment registers are used.

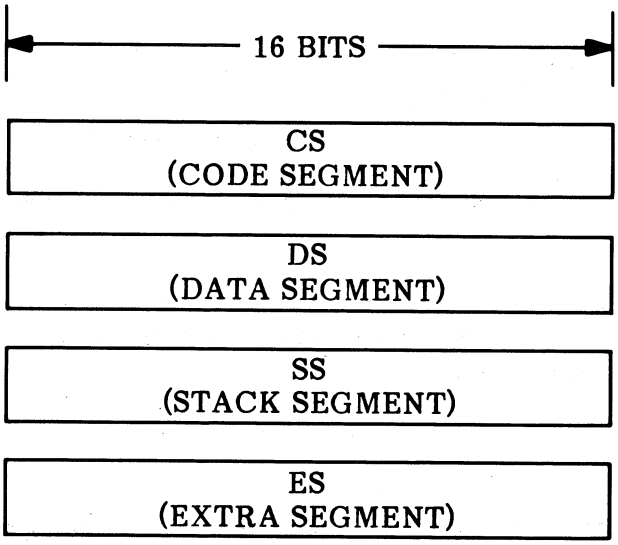


Figure A-6. Segment Registers

OPERATION

RESET

The 8088 is reset with a positive pulse on the reset line (pin 21 of the processor). The processor terminates all activity on the positive edge of the pulse and initializes to the state shown in figure A-7 on the falling edge of the pulse.

CPU COMPONENT	INITIALIZED STATE (HEX)
FLAGS	CLEAR
INSTRUCTION POINTER	0000
CS REGISTER	FFFF
DS REGISTER	0000
SS REGISTER	0000
ES REGISTER	0000
QUEUE	EMPTY

Figure A-7. RESET Initialization States

Upon initialization, the code segment register contains FFFF hex and the instruction pointer, 0000. This points the processor to absolute memory location FFFF0 hex. Since this location is only 16 bytes from the top of memory, it usually contains an intersegment jump to the start of the program.

Segmentation

The 8088 provides 20 address lines to access up to one megabyte of memory. The megabyte is addressed by the CPU using a segmented addressing scheme, rather than the traditional linear addressing used by most 8-bit microprocessors. Segmented addressing permits the CPU to generate a 20-bit address from a 16-bit value.

A segment is a contiguous block of memory up to 64K bytes long. The CPU can address up to four such segments at a time by maintaining the starting address for each. The segment starting addresses are maintained in segment registers. All segments begin on 16-byte boundaries. There are no other restrictions on segment locations. Relative to each other, segments may be contiguous, partially overlapped, fully overlapped, or separated. This is shown in figure A-8.

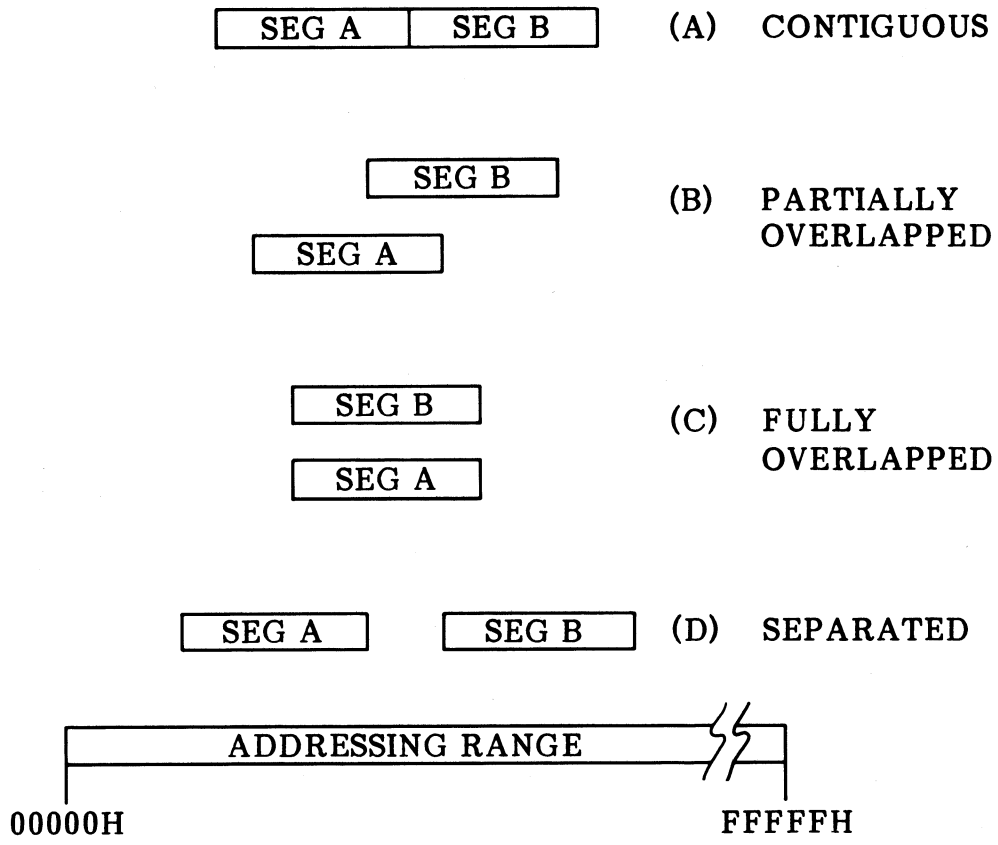


Figure A-8. Segment Mapping

The 8088 uses three of the four segments to contain programs, data, and stack. The fourth segment is an extra segment that typically is used to store data. Each of these four memory spaces is assigned a segment register. Each segment register points to the base address of the corresponding segment of memory. The CS register points to the current code segment; instructions are fetched from this segment. The SS register points to the current stack segment; stack operations are performed on locations in this segment. The DS register points to the current data segment; it generally contains program variables. The ES register points to the current extra segment, which also typically is used for data storage. This is shown in figure A-6, page A-6.

Memory Structure

Twenty bits of address are required to access any location within the 8088 megabyte addressing space. The 20 bits are composed of two values called the base and the offset.

The base address is the starting address of the segment. The upper 16 bits of this address are contained within a segment register. The lower 4 bits are implied zeros. This explains the placement restriction of segment starting addresses on 16-byte boundaries. If, for example, a segment register were to contain the value 1234 hex, the memory address for the start of that segment would be 12340 hex. This is shown in figure A-9.

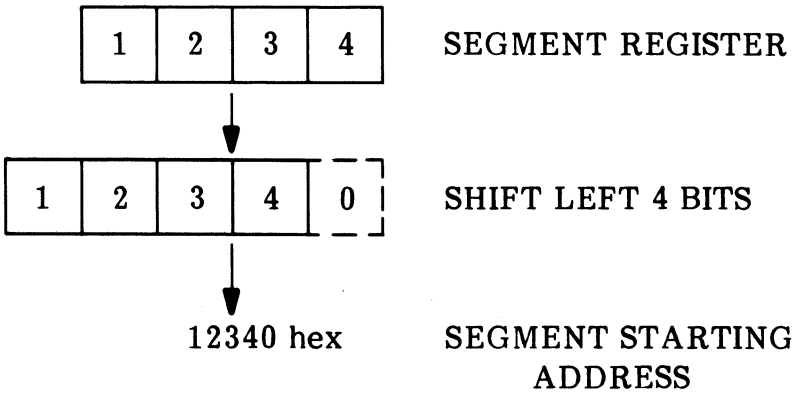


Figure A-9. Segment Starting Address

Once the segment starting address is obtained, a 16-bit offset is required to address anywhere within the 64K segment. An offset is automatically added to the segment starting address prior to every memory access. For an instruction fetch, the CS register is chosen as the segment base and the IP register added as an offset. For a stack operation, the SS is chosen as the segment base and the SP register added as an offset. Most data is assumed to reside in the current data segment. The offset is a value calculated by the execution unit (EU). This value is termed "effective address." The effective address calculation is based on the addressing mode specified in the instruction. The 8088 memory addressing components are summarized in figure A-10.

TYPE OF MEMORY REFERENCE	DEFAULT SEGMENT	ALTERNATE SEGMENT	OFFSET ADDRESS
Instruction Fetch	CS	NONE	IP
Stack Operation	SS	NONE	SP
String Source	DS	CS,ES,SS	SI
String Destination	ES	NONE	DI
BP Used as Base Register	SS	CS,DS,ES	Effective Address
General Data Read/Write	DS	CS,ES,SS	Effective Address

Figure A-10. 8088 Address Components

The 20-bit address can be formed by adding the the base address and offset together. The CPU automatically performs this operation. The programmer must ensure that: the segment register is loaded with a value that will point to the correct segment base, and the instruction uses the correct offset.

I/O Structure

The 8088 can directly address up to 64K I/O ports. Because this requires only 16 bits of address, there is no need to segment the I/O space. Therefore, all ports can be considered to be in one segment. The first 256 ports are directly addressable. The entire 64K ports are indirectly addressable.

Interrupts

The 8088 is capable of recognizing both internally and externally generated interrupts. The sources of interrupts are shown in figure A-11. Regardless of the interrupting source, it is identified to the CPU by a type number. The type number is used as a pointer to an interrupt vector table where the physical address of the device's interrupt service routine is located.

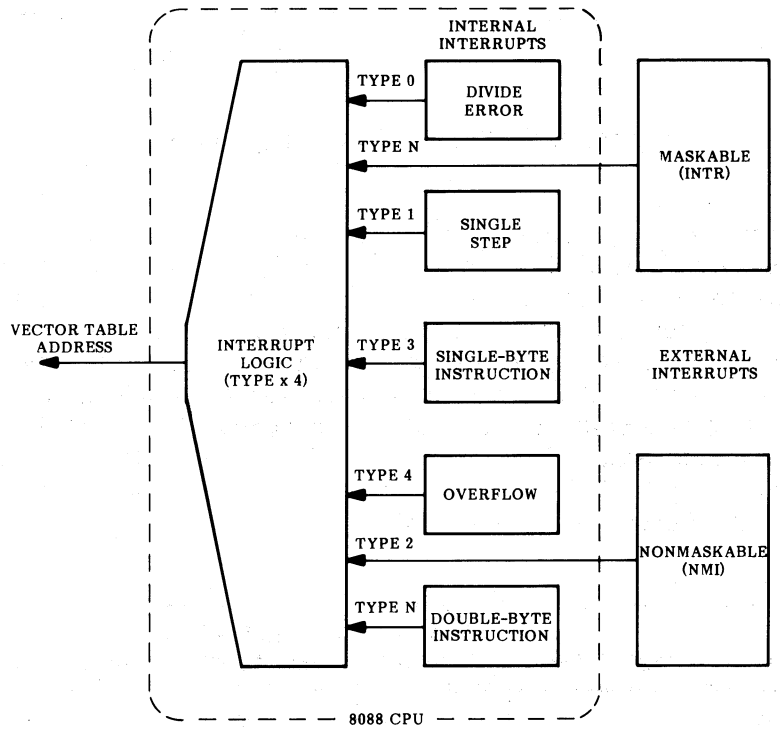


Figure A-11. 8088 Interrupt Sources

The type number is an 8-bit value. This means there can be up to 256 uniquely identified interrupting sources. As seen in figure A-11, some of the interrupting sources have predefined type numbers and some do not. Those that do not are shown providing a type number "N," where N is an 8-bit value from 0 to 255.

Externally Generated Interrupts: As mentioned previously, interrupts can be generated externally or internally to the processor. The external interrupts are maskable interrupt request (INTR) or nonmaskable interrupt request (NMI).

External Interrupt Request (INTR)

The response of the CPU to an active INTR depends on the state of the interrupt flag (I). If I is reset (logical zero), the CPU ignores INTR and continues execution with the next instruction. If I is set (logical one), the CPU completes execution of the current instruction and then processes the interrupt. The CPU acknowledges INTR by executing two consecutive interrupt acknowledge bus cycles. The first cycle is used to signal the interrupting device that the request has been honored. The second interrupt acknowledge cycle is used by the interrupting device to gate a type number "N" onto the data bus.

Nonmaskable Interrupt Request (NMI)

An active NMI is not maskable. NMI is an edge-triggered signal usually used to signal catastrophic events. Interrupt requests arriving on NMI are latched by the CPU and have a higher priority than an interrupt request on INTR. Nonmaskable interrupt requests have a predefined type number of 2. The processor does not need a type number and does not execute the interrupt acknowledge bus cycles in response to NMI.

Internally Generated Interrupts: There are several sources of internally generated interrupts, as seen in figure A-11. Internally generated interrupts are caused by the following conditions:

A Software Interrupt Instruction

There are two forms for the software INT instruction, as indicated in figure A-11. The first form is a single-byte instruction. The single-byte instruction has an implied type number of 3. The double-byte instruction has a type number defined by the second byte of the instruction. This means that the second byte of the double-byte INT instruction is a hex value from 00 to FF that defines the type number.

Division by zero

When an arithmetic divide instruction is executed with a zero divisor, a type 0 interrupt is generated.

Single-Step

If the trap flag of the I register is set, a type 1 interrupt is generated following the execution of each instruction.

An arithmetic overflow

If the overflow flag of the I register is set, a type 4 interrupt is generated upon execution of the "interrupt on overflow" (INTO) instruction.

Once the interrupt logic of figure A-11, page A-10, has the type number from the interrupting source, the following events occur:

1. The CPU flag register is pushed onto the stack. This ensures that the processor can be restored to its current state after the interrupt has been processed.

2. The contents of the CS and IP are pushed onto the stack. This ensures that the processor will return and execute the correct instruction after the interrupt has been processed.

3. The trap flag (T) and interrupt flag (I) are cleared. This operation prevents additional interrupts from occurring until the return from interrupt (IRET) is executed or until the program specifically re-enables the interrupt with the set interrupt (STI) instruction.

4. The CPU forms a pointer to an interrupt vector table by multiplying the type number by four. The reason for multiplying the type number by four is to leave enough room to store the physical address (code segment and instruction pointer) of the interrupt service routine for each table entry. The interrupt vector table is shown in figure A-12.

5. The first word from the vector table is loaded into the IP register. The second word is loaded into the CS register. Program execution is subsequently transferred to the service routine.

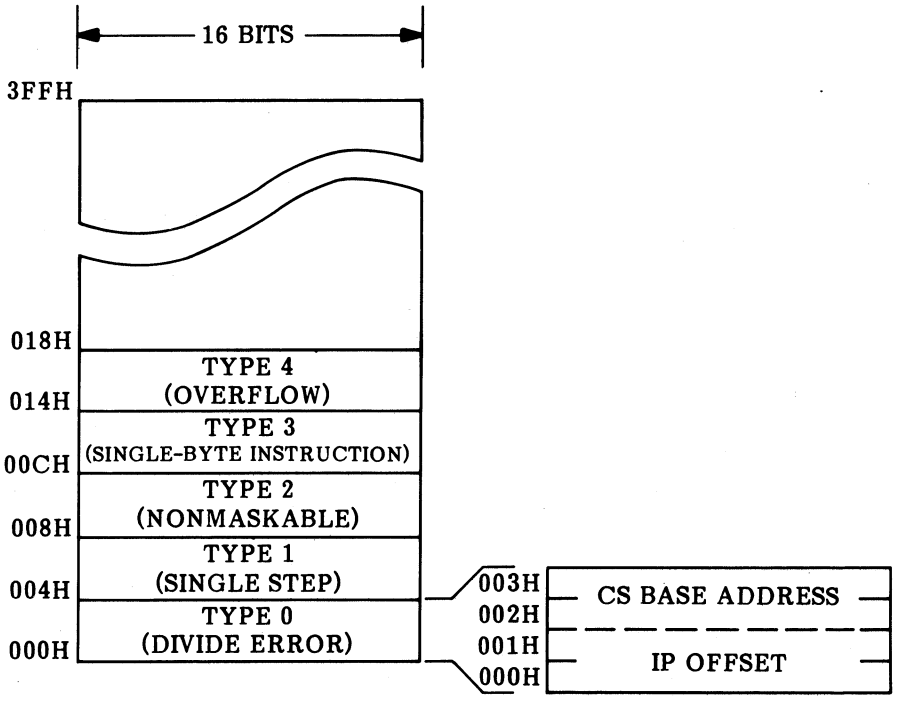
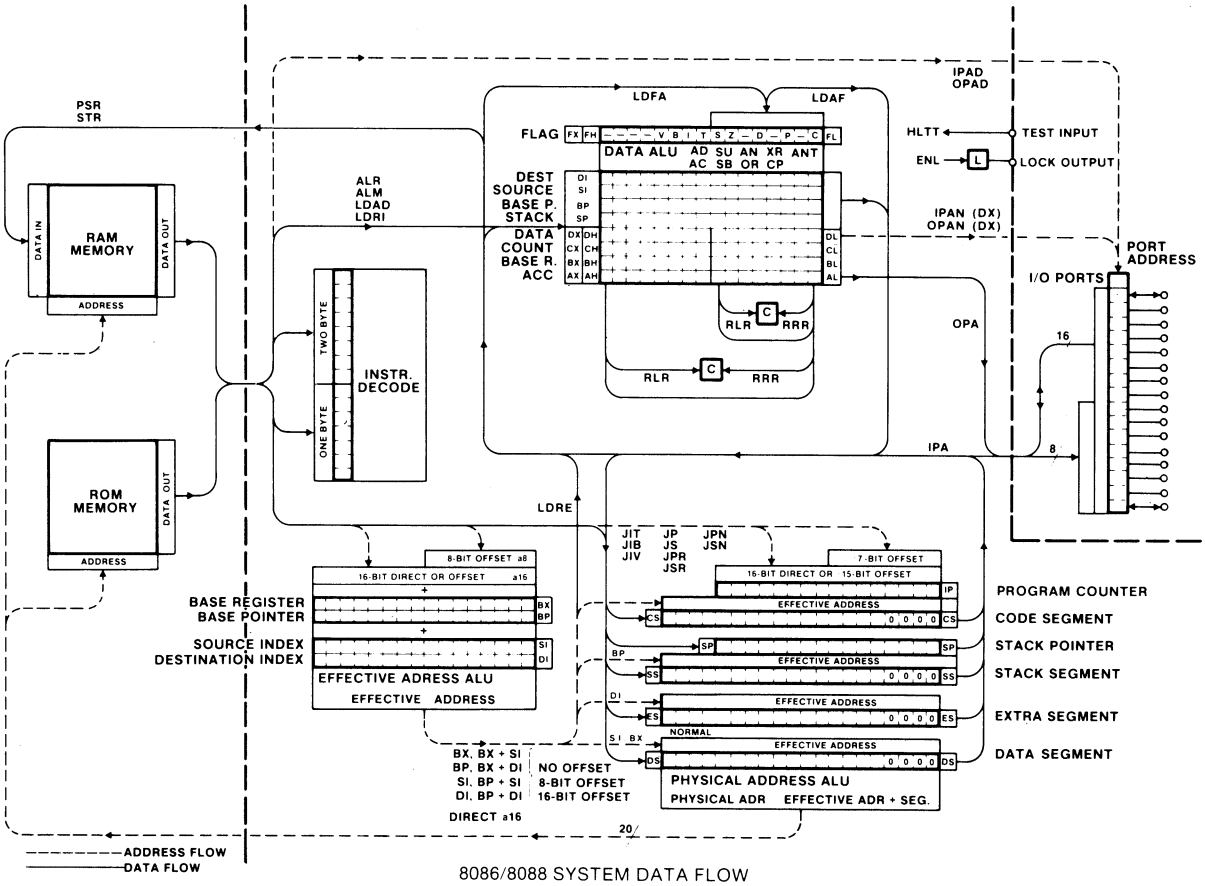


Figure A-12. Interrupt Vector Table

The last instruction of the interrupt service routine will be the IRET instruction, which returns the processor to the state from which it was interrupted.

System Data Flow

A conceptual drawing of the 8088 architecture is shown in figure A-13. The drawing shows all of the resources available to the user from the 8088 microprocessor. Data flow affected by instructional operations is represented by solid flow lines. Addressing of memory and I/O is represented by broken flow lines.



8086/8088 SYSTEM DATA FLOW

Figure A-13. 8086/8088 System Data Flow

INTRODUCTION

This appendix contains the necessary information to program the 8088 using the STD mnemonics subset.

THE 8088 INSTRUCTION SUBSET

The 8088 is a complex microprocessor to program. To simplify the programming task, this appendix presents a subset of the 8088 instruction set. This subset is in STD mnemonics and is very similar to the 8085 instruction set. The major differences between the 8085 and 8088 instruction sets are due to the 8088 segmentation and relative jumps. Figure B-1 shows the 8085 register subset as used for the 8088 instruction subset.

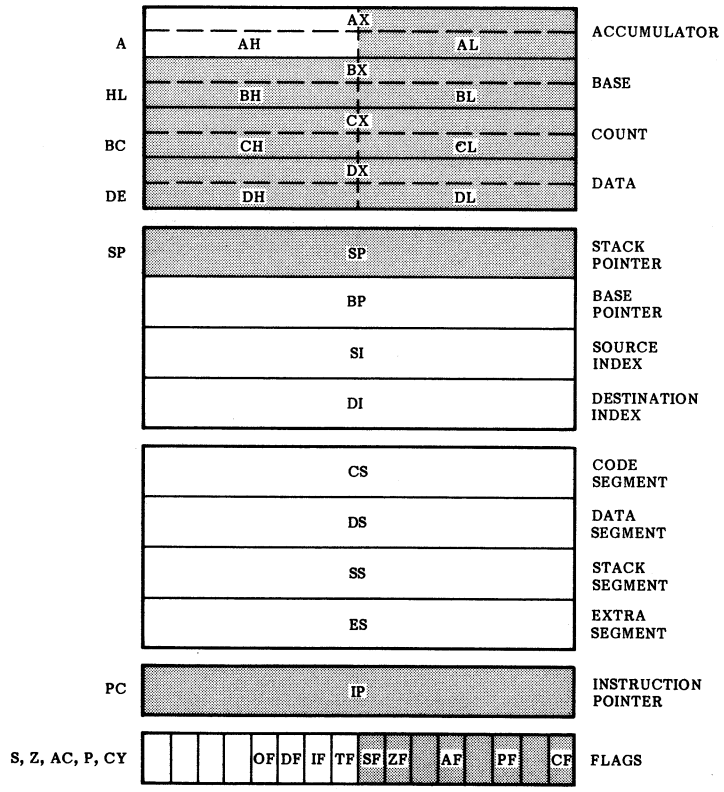


Figure B-1. 8085 Register Subset (Shaded)

Descriptions of the 8088 STD mnemonics are on the following pages. Following the descriptions are instruction tables containing the op-codes of all the instructions.

STD INSTRUCTION MNEMONICS

The STD Instruction Mnemonics are a structured set of abbreviations for assembly language programming. The mnemonic abbreviations are tailored to be common for all microprocessor instruction sets. The structure of the mnemonics is designed to aid recognition and memory, to minimize keystrokes for entry, and to assist in software automation. The STD Mnemonics are standard in that they do not change but keep the same meaning for all processors. They are structured in that they are derived from a set of simple rules.

The instruction mnemonic is an abbreviated action statement containing an operator, a locator, and a qualifier, plus a supplemental and separate modifier.

1. The operator is a unique two-letter abbreviation of the word that states the action.

2. The locator follows the operator and designates the operand or data to be operated on. Instructions without operands ignore the locator.

3. The qualifier states the addressing mode or provides further qualifying information for compound instructions.

4. The modifier states the variable operand information such as registers, labels, conditions, addresses, and data.

The operator, locator, and qualifier letters are strung together to form the instruction mnemonic. The modifier, when needed, stands alone either in its own separate column or separated by spaces or additional lines.

	OPERATOR	LOCATOR	QUALIFIER	MODIFIER	INSTRUCTION DESCRIPTION
RTS	RT	S			Return from Subroutine
CLA	CL	A			Clear A
LDAD	LD	A	D		Load A Direct
LDA B	LD	A		B	Load A with B
LDAN (BC)	LD	A	N	(BC)	Load A Indirect using BC as an Address Pointer
JSR (LABEL)	JS		R	(LABEL)	Jump to Subroutine Located at (LABEL)

Figure B-2. Examples of Instruction Mnemonic Structure

For further information on the complete 8088 instruction set, see the following references:

iAPX 88 Book, with an Introduction to the 188. Reston Publishing Company, Inc., Reston, Virginia. #R3016-4.

iAPX 86/88, 186/188 User's Manual — Programmer's Reference. Reston Publishing Company, Inc., Reston, Virginia. #R3035-4.

OPERATIONS

AC	ADD WITH CARRY
AD	ADD
AJ	ADJUST (DECIMAL)
AN	AND
CL	CLEAR
CM	COMPLEMENT
CP	COMPARE
DC	DECREMENT
DS	DISABLE
EN	ENABLE
HLT	HALT
IC	INCREMENT
IP	INPUT FROM PORT
JI	JUMP TO INTERRUPT
JP	JUMP
JS	SUBROUTINE JUMP
LD	LOAD
NOP	NO OPERATION
OP	OUTPUT TO PORT
OR	(INCLUSIVE) OR
PL	PULL VIA STACK
PS	PUSH VIA STACK
RL	ROTATE LEFT
RR	ROTATE RIGHT
RS	RESET
RTS	RETURN FROM SUBROUTINE
SB	SUBTRACT WITH BORROW
SE	SET
ST	STORE
SU	SUBTRACT
XC	EXCHANGE
XR	EXCLUSIVE OR

QUALIFIERS

C	WITH CARRY
D	DIRECT ADDRESSING, DECIMAL
I	IMMEDIATE ADDRESSING
N	INDIRECT ADDRESSING
R	RELATIVE
L	LONG, 20-BIT
S	SHORT, 8-BIT (±127)

LOCATORS, MODIFIERS

A	ACCUMULATOR REGISTER
B,C D,E H,L	GENERAL 8-BIT REGISTERS
F	FLAG REGISTER
AF	ACCUMULATOR, FLAGS PAIRED
BC DE HL	GENERAL REGISTER PAIRS (16-BIT DATA, POINTERS)
P	ANY REGISTER PAIR
R	ANY SINGLE REGISTER
M	MEMORY, ADDRESSED INDIRECTLY
I	INTERRUPT or INTERRUPT MASK
PC	PROGRAM COUNTER
SP	STACK POINTER
CS	CODE SEGMENT REGISTER
DS	DATA SEGMENT REGISTER
SS	STACK SEGMENT REGISTER
x	register (B,C,D,E,H,L) accumulator (A)
xx	register pair (AF,BC,DE,HL,SP)
d	8-bit data
dd	16-bit data
aa	16-bit address
r	8-bit displacement
rr	16-bit displacement
ss	16-bit segment value
p	8-bit I/O port address
f	Flag condition to be tested (UN,C0,Z0,S0,P0,C1,Z1,S1,P1)
c	Carry Flag
d	Decimal Flag
z	Zero Flag
s	Sign Flag (MSB)
p	Parity Flag
1,0	State of Indicated Flag
H	High byte of operand
L	Low byte of operand
(P)	Indirect

DEFINITIONS

OFFSET = positive number of bytes from the start of the segment

DISPLACEMENT = signed number of bytes from the current instruction pointer (relative addressing)

FORMAT: ACAx mod

OPERATION: A <- c + A + mod

FLAGS: c,d,z,s,p

DESCRIPTION: Add the modifier and carry flag to A.

INTEL MNEMONIC(*): ADC

MNEMONIC	MODIFIER	DESCRIPTION
ACA	R	Add R to A with Carry
ACAN	(HL)	Add (HL) to A with Carry
ACAI	d	Add d to A with Carry

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
ACA	x	12C0	12C5	12C1	12C6	12C2	12C7	12C3	1207	14d	c,d,z,s,p	Add x to Acc with Carry

(*) Intel Corporation mnemonics copyright 1984

FORMAT: ADAx mod

OPERATION: A <- A + mod

FLAGS: c,d,z,s,p

DESCRIPTION: Add the modifier to A

INTEL MNEMONIC(*): ADD

MNEMONIC	MODIFIER	DESCRIPTION
ADA	R	Add R to A
ADAN	(HL)	Add (HL) to A
ADAI	d	Add d to A
ADP	P1,P2	Add P2 to P1

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
ADA	x	02C0	02C5	02C1	02C6	02C2	02C7	02C3	0207	04d	c,d,z,s,p	Add x to Acc

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
ADP	BC,xx		03C9	03CA	03CB	03CC	c,d,z,s,p	Add Pair xx to BC
ADP	DE,xx		03D1	03D2	03D3	03D4	c,d,x,s,p	Add Pair xx to DE
ADP	HL,xx		03D9	03DA	03DB	03DC	c,d,z,s,p	Add Pair xx to HL
ADP	SP,xx		03E1	03E2	03E3	03E4	c,d,z,s,p	Add Pair xx to SP

(*) Intel Corporation mnemonics copyright 1984

FORMAT: AJAD

OPERATION: If LSN of A > 9 or D1 then

A <- A + 6, and D1

If A > 9FH or C1 then

A <- A + 60H, and C1

FLAGS: c,d,z,s,p

DESCRIPTION: The result of adding two packed decimal operands is adjusted to form two four-bit binary coded decimal digits.

INTEL MNEMONIC(*): DAA

MNEMONIC	MODIFIER	DESCRIPTION
AJAD		Adjust A Decimally

INSTR	QUAL MOD	A		FLAG STATUS	DESCRIPTION OF OPERATION
AJAD		27	A(adj)->A	c,d,z,s,p	Adjust A decimally

(*) Intel Corporation mnemonics copyright 1984

FORMAT: ANAx mod

OPERATION: A <- A AND mod

FLAGS: C0,d,z,s,p

DESCRIPTION: AND the modifier with A.

INTEL MNEMONIC(*): AND

MNEMONIC	MODIFIER	DESCRIPTION
ANA	R	AND R with A
ANAN	(HL)	AND (HL) with A
ANAI	d	AND d with A

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
ANA	x	22C0	22C5	22C1	22C6	22C2	22C7	22C3	2207	24d	C0,d,z,s,p	And x with Acc

(*) Intel Corporation mnemonics copyright 1984

FORMAT: CLAC

OPERATION: A <- 0, c <- 0

FLAGS: C0,D0,Z1,S0,P1

DESCRIPTION: Clear A and carry flag

INTEL MNEMONIC(*): XOR AL, AL

MNEMONIC	MODIFIER	DESCRIPTION
CLAC		Clear A and c

INSTR	QUAL MOD	A		FLAG STATUS	DESCRIPTION OF OPERATION
CLAC		32C0	0->A,0->C	C0,D0,Z1,S0,P1	Clear Acc and Carry

(*) Intel Corporation mnemonics copyright 1984

FORMAT: CMx

OPERATION: x <- not x

FLAGS: none

DESCRIPTION: Each bit is inverted to form the one's complement

INTEL MNEMONIC(*): NOT

MNEMONIC	MODIFIER	DESCRIPTION
CMA		Complement A
CMR	R	Complement R
CMMN	(HL)	Complement (HL)

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
CMA		F6D0										Complement Acc
CMR	x		F6D5	F6D1	F6D6	F6D2	F6D7	F6D3				Complement Register
CMMN									F617			Complement Memory Indirect

(*) Intel Corporation mnemonics copyright 1984

FORMAT: CMC

OPERATION: If C0 then C1
If C1 then C0

FLAGS: c

DESCRIPTION: If the carry flag is reset, CMC sets it to one. If the carry flag is set, CMC resets it to zero.

INTEL MNEMONIC(*): CMC

MNEMONIC	MODIFIER	DESCRIPTION
CMC		Complement Carry

INSTR	QUAL MOD	A		FLAG STATUS	DESCRIPTION OF OPERATION
CMC		F5	C->C	c	Complement Carry

(*) Intel Corporation mnemonics copyright 1984

FORMAT: CPAx mod

OPERATION: A - mod

FLAGS: c,d,z,s,p

DESCRIPTION: The modifier is subtracted from A. The result is not saved, but the flags are updated.

INTEL MNEMONIC(*): CMP

MNEMONIC	MODIFIER	DESCRIPTION
CPA	R	Compare R with A
CPAN	(HL)	Compare (HL) with A
CPAI	d	Compare d with A

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
CPA	x	3AC0	3AC5	3AC1	3AC6	3AC2	3AC7	3AC3	3A07	3Cd	c,d,s,z,p	Compare x with Acc

(*) Intel Corporation mnemonics copyright 1984

FORMAT: DCx mod

OPERATION: x <- x-1

FLAGS: d,z,s,p

DESCRIPTION: The specified register is decremented by one.

INTEL MNEMONIC(*): DEC

MNEMONIC	MODIFIER	DESCRIPTION
DCA		Decrement A
DCR	R	Decrement R
DCMN	(HL)	Decrement (HL)
DCP	P	Decrement P

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
DCA		FEC8									d,z,s,p	Decrement Acc
DCR	x		FECD	FEC9	FECE	FECA	FECF	FECB			d,z,s,p	Decrement Register x
DCMN	(HL)								FEOF		d,z,s,p	Decrement Memory

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
DCP	xx		49	4A	4B	4C	d,z,s,p	Decrement Pair xx

(*) Intel Corporation mnemonics copyright 1984

FORMAT: DSI

OPERATION: Disable Interrupts

FLAGS: none

DESCRIPTION: The interrupt control bit is cleared to a zero, masking the external interrupt request (INTRQ*).

INTEL MNEMONIC(*): CLI

MNEMONIC	MODIFIER	DESCRIPTION
DSI		Mask INTRQ*

INSTR	QUAL MOD			DESCRIPTION OF OPERATION
DSI		FA		Disable Interrupt Structure

FORMAT: ENI

OPERATION: Enable Interrupts

FLAGS: none

DESCRIPTION: The interrupt control bit is set to a one, enabling the external interrupt request (INTRQ*).

INTEL MNEMONIC(*): STI

MNEMONIC	MODIFIER	DESCRIPTION
ENI		Enable INTRQ*

INSTR	QUAL MOD			DESCRIPTION OF OPERATION
ENI		FB		Enable Interrupt Structure

(*) Intel Corporation mnemonics copyright 1984

FORMAT: ENS segment

DESCRIPTION: The modifier specifies the segment for the following instruction to use for memory accesses.

INTEL MNEMONIC(*): CS:, DS:

MNEMONIC	MODIFIER	DESCRIPTION
ENS	CS	Enable Code Segment
ENS	DS	Enable Data Segment

INSTR	MOD			DESCRIPTION OF OPERATION
ENS	CS	2E		Enable Code Segment Access
ENS	DS	3E		Enable Data Segment Access

(*) Intel Corporation mnemonics copyright 1984

FORMAT: HLT

OPERATION: none

FLAGS: none

DESCRIPTION: The CPU enters a halt state. The CPU exits the halt state upon reset, NMIRQ*, or INTRQ* if the interrupts are enabled.

INTEL MNEMONIC(*): HLT

MNEMONIC	MODIFIER	DESCRIPTION
HLT		Halt

INSTR	MOD			DESCRIPTION OF OPERATION
HLT		F4		Halt

(*) Intel Corporation mnemonics copyright 1984

FORMAT: HLTT

OPERATION: none

FLAGS: none

DESCRIPTION: The CPU enters a wait state if the signal on the TEST* pin is not asserted. The wait state is cleared and the program execution continues when the TEST pin is asserted.

INTEL MNEMONIC(*): WAIT

MNEMONIC	MODIFIER	DESCRIPTION
HLTT		Halt and wait for TEST* active

INSTR	MOD			DESCRIPTION OF OPERATION
HLTT		9B		Halt until TEST* low

(*) Intel Corporation mnemonics copyright 1984

FORMAT: ICx

OPERATION: x <- x + 1

FLAGS: d,z,s,p

DESCRIPTION: The specified register is incremented by one.

INTEL MNEMONIC(*): INC

MNEMONIC	MODIFIER	DESCRIPTION
ICA		Increment A
ICR	R	Increment R
ICMN	(HL)	Increment (HL)
ICP	P	Increment P

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
ICA		FEC0									d,z,s,p	Increment Acc
ICR	x		FEC5	FEC1	FEC6	FEC2	FEC7	FEC3			d,z,s,p	Increment Register x
ICMN	(HL)								FE07		d,z,s,p	Increment Memory

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
ICP	xx		41	42	43	44	d,z,s,p	Increment Pair xx

(*) Intel Corporation mnemonics copyright 1984

FORMAT: IPAx mod

OPERATION: A <- (port)

FLAGS: none

DESCRIPTION: A byte of data is transferred from the specified port into the accumulator.

INTEL MNEMONIC(*): IN

MNEMONIC	MODIFIER	DESCRIPTION
IPA	p	Input A Direct
IPAN	(DE)	Input A Indirect

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
IPA		E4p						Input to A from Port p
IPAN	(DE)	EC						Input to A Indirect

(*) Intel Corporation mnemonics copyright 1984

FORMAT: JI mod

OPERATION:

```
    SP    <- SP - 2
(SP + 1)  <- FH
  (SP)    <- FL
    i     <- 0
    SP    <- SP - 2
(SP + 1)  <- CSH
  (SP)    <- CSL
   CSH    <- (mod * 4 + 3)
   CSL    <- (mod * 4 + 2)
    SP    <- SP - 2
(SP + 1)  <- PCH
  (SP)    <- PCL
   PCH    <- (mod * 4 + 1)
   PCL    <- (mod * 4)
```

FLAGS: none

DESCRIPTION: Activates the interrupt procedure. The stack pointer is decremented by two, the flags are pushed, and the interrupt control bit is cleared. The code segment is pushed onto the stack. The address of the interrupt pointer is calculated by multiplying the mod by four. The second word of the interrupt pointer replaces the code segment. The program counter is pushed onto the stack and is replaced by the first word of the interrupt pointer.

INTEL MNEMONIC(*): INT

MNEMONIC	MODIFIER	DESCRIPTION
JI	d	Jump to Interrupt

INSTR	MOD			DESCRIPTION OF OPERATION
JI	d	CDd		Software Interrupt at Vector d
JI	3	CC		Software Interrupt at Vector 3

(*) Intel Corporation mnemonics copyright 1984

FORMAT: JPx leftmod, rightmod

OPERATION:

Short relative	Relative
if condition = true, then PC <- PC + disp8	PC <- PC + disp16
Long	**Indirect**
PC <- adr32 offset CS <- adr32 segment	PC <- (P)

FLAGS: none

DESCRIPTION: The destination address is placed in PC (and CS for a long jump).

INTEL MNEMONIC(*): JMP

MNEMONIC	MODIFIER	DESCRIPTION
JPRS	condition,disp8	Jump Relative on Condition to (PC+r)
JPR	disp16	Jump Relative to (PC+rr)
JPL	adr32	Jump Long to adr32
JPN	(P)	Jump Indirect to (P)

FLAG CONDITIONS (f)											DESCRIPTON OF OPERATION
			≥	≠	+	ODD	<	=	-	EVEN	ARITHMETIC
INSTR	MOD	UN	C0	Z0	S0	P0	C1	Z1	S1	P1	LOGICAL
JPL	UN	EAaass									Jump Long
JPN	(BC)	FFE1									Jump Indirect, BC = 16-bit offset
JPN	(DE)	FFE2									Jump Indirect, DE = 16-bit offset
JPN	(HL)	FFE3									Jump Indirect, HL = 16-bit offset
JPR	UN	E9rr									Jump Relative, 16-bit Displacement
JPRS	f	EBr	73r	75r	79r	7Br	72r	74r	78r	7Ar	Jump Short on Condition f, 8-bit displacement

(*) Intel Corporation mnemonics copyright 1984

FORMAT: JSx mod

OPERATION:

If LONG then

```
     SP  <-  SP - 2
(SP + 1) <-  CSH
   (SP)  <-  CSL
     CS  <-  DEST CS
     SP  <-  SP - 2
(SP + 1) <-  PCH
   (SP)  <-  PCL
     PC  <-  DEST PC
```

If RELATIVE then

```
     SP  <-  SP-2
(SP + 1) <-  PCH
   (SP)  <-  PCL
     PC  <-  PC + DISP
```

FLAGS: none

DESCRIPTION: The program counter (and code segment if a long jump) is saved on the stack. The subroutine address is placed in PC and CS.

INTEL MNEMONIC(*): CALL

MNEMONIC	MODIFIER	DESCRIPTION
JSR JSL	displacement address	PC <- PC + displacement PC <- address offset CS <- address segment

FLAG CONDITIONS (f)											DESCRIPTON OF OPERATION
			≥	≠	+	ODD	<	=	-	EVEN	ARITHMETIC
INSTR	MOD	UN	C0	Z0	S0	P0	C1	Z1	S1	P1	LOGICAL
JSL JSR	UN UN	9Aaass E8rr									Jump Subroutine Long Jump Subroutine, 16-bit Displacement

(*) Intel Corporation mnemonics copyright 1984

FORMAT: LDx leftmod, rightmod

OPERATION: leftmod <- rightmod

FLAGS: none

DESCRIPTION: The contents of the right modifier are placed into the left modifier.

INTEL MNEMONIC(*): MOV

MNEMONIC	MODIFIER	DESCRIPTION
LDA	R	Load A with R
LDR	B,R	Load B with R
LDR	C,R	Load C with R
LDR	D,R	Load D with R
LDR	E,R	Load E with R
LDR	H,R	Load H with R
LDR	L,R	Load L with R
LDMNI	(HL)	Load (HL) Immediate
LDAD		Load A Direct
LDAI	d	Load A Immediate
LDAN	(HL)	Load A Indirect
LDRI	R,d	Load R Immediate
LDRN	R,(HL)	Load R Indirect
LDP	P,CS	Load P with CS
LDP	P,DS	Load P with DS
LDP	P,SS	Load P with SS
LDP	DS,P	Load DS with P
LDP	SP,P	Load SP with P
LDP	SS,P	Load SS with P
LDPI	P,dd	Load P Immediate
LDPD	P,aa	Load P Direct

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
LDA	x	8AC0	8AC5	8AC1	8AC6	8AC2	8AC7	8AC3	8A07	B0d		
LDR	B,x	8AE8	8AED	8AE9	8AEE	8AEA	8AEF	8AEB	8A2F	B5d		
LDR	C,x	8AC8	8ACD	8AC9	8ACE	8ACA	8ACF	8ACB	8A0F	B1d		
LDR	D,x	8AF0	8AF5	8AF1	8AF6	8AF2	8AF7	8AF3	8A37	B6d		Load with Register x
LDR	E,x	8AD0	8AD5	8AD1	8AD6	8AD2	8AD7	8AD3	8A17	B2d		
LDR	H,x	8AF8	8AFD	8AF9	8AFE	8AFA	8AFF	8AFB	8A3F	B7d		
LDR	L,x	8AD8	8ADD	8AD9	8ADE	8ADA	8ADF	8ADB	8A1F	B3d		
LDMNI	(HL),d									C607d		Load Memory Immediate
LDAD		A0aa										Load Acc Direct

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
LDP	xx,CS		8CC9	8CCA	8CCB	8CCC		Load pair with CS
LDP	xx,DS		8CD9	8CDA	9CDB	8CDC		Load pair with DS
LDP	xx,SS		8CD1	8CD2	8CD3	8CD4		Load pair with SS
LDP	DS,xx		8ED9	8EDA	8EDB	8EDC		Load DS with pair
LDP	SS,xx		8ED1	9ED2	8ED3	8ED4		Load SS with pair
LDP	SP,xx		8BE1	8BE2	8BE3	8BE4		Load SP with pair
LDPI	xx		C7C1dd	C7C2dd	C7C3dd	C7C4dd		Load Pair Immediate
LDPD	xx		8B0Eaa	8B16aa	8B1Eaa	8B26aa		Load Pair Direct

(*) Intel Corporation mnemonics copyright 1984

FORMAT: NOP

OPERATION: none

FLAGS: none

DESCRIPTION: No operation is performed.

INTEL MNEMONIC(*): NOP

MNEMONIC	MODIFIER	DESCRIPTION
NOP		No operation

INSTR	MOD		DESCRIPTION OF OPERATION
NOP		90	No operation

(*) Intel Corporation mnemonics copyright 1984

FORMAT: OPAx mod

OPERATION: (port) <- A

FLAGS: none

DESCRIPTION: The contents of the accumulator is transferred to the specified port.

INTEL MNEMONIC(*): OUT

MNEMONIC	MODIFIER	DESCRIPTION
OPA	p	Output A Direct
OPAN	(DE)	Output A Indirect

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
OPA		E6p						Output from A to Port p
OPAN	(DE)	EE						Output from A Indirect

(*) Intel Corporation mnemonics copyright 1984

FORMAT: ORAx mod

OPERATION: A <- A OR mod

FLAGS: C0,d,z,s,p

DESCRIPTION: A is logically ORed with the modifier. The result is placed in A. The carry flag is reset.

INTEL MNEMONIC(*): OR

MNEMONIC	MODIFIER	DESCRIPTION
ORA	R	OR R with A
ORAN	(HL)	OR (HL) with A
ORAI	d	OR d with A

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
ORA	x	0AC0	0AC5	0AC1	0AC6	0AC2	0AC7	0AC3	0A07	0Cd	C0,d,z,s,p	OR x with Acc

(*) Intel Corporation mnemonics copyright 1984

FORMAT: PLP mod

OPERATION:
```
(modL)  <- (SP)
(modH)  <- (SP + 1)
   SP   <- SP + 2
```

FLAGS: none

DESCRIPTION: Transfer a word from the top of stack to the specified register pair.

INTEL MNEMONIC(*): POP

MNEMONIC	MODIFIER	DESCRIPTION
PLP	P	Pull stack to P

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
PLP	xx	9D58	59	5A	5B	5C	(if AF:c,d,z,s,p)	Pull Pair from Stack

(*) Intel Corporation mnemonics copyright 1984

FORMAT: PSP mod

OPERATION:
```
    SP   <- SP - 2
(SP + 1) <- (modH)
   (SP)  <- (modL)
```

FLAGS: none

DESCRIPTION: Decrements the stack pointer by two and then transfers a word from the specified register pair to the current top of stack.

INTEL MNEMONIC(*): PUSH

MNEMONIC	MODIFIER	DESCRIPTION
PSP	P	Push P on stack

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
PSP	xx	509C	51	52	53	54		Push Pair to Stack

(*) Intel Corporation mnemonics copyright 1984

FORMAT: RLAx

OPERATION: without carry: with carry:

FLAGS: c

DESCRIPTION: Rotate A left.

INTEL MNEMONIC(*): RCL, ROL

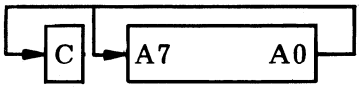
MNEMONIC	MODIFIER	DESCRIPTION
RLA RLAC		Rotate A left Rotate A left with Carry

INSTR	QUAL MOD	A		FLAG STATUS	DESCRIPTION OF OPERATION
RLA RLAC		D0C0 D0D0	A7->A0->A1...;A7->C A7->C-> A0->A1...	c c	Rotate Acc Left Rotate Acc Left with Carry

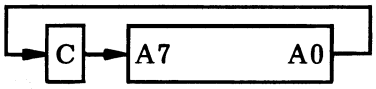
(*) Intel Corporation mnemonics copyright 1984

FORMAT: RRAx

OPERATION: Without carry:



With carry:



FLAGS: c

DESCRIPTION: Rotate A right.

INTEL MNEMONIC(*): RCR, ROR

MNEMONIC	MODIFIER	DESCRIPTION
RRA		Rotate A right
RRAC		Rotate A right with Carry

INSTR	QUAL MOD	A		FLAG STATUS	DESCRIPTION OF OPERATION
RRA		D0C8	A0->C;A0->A7->A6	c	Rotate Acc Right
RRAC		D0D8	A0->C->A7->A6...	c	Rotate A Right with Carry

(*) Intel Corporation mnemonics copyright 1984

FORMAT: RSC

OPERATION: c <- 0

FLAGS: C0

DESCRIPTION: The carry flag is reset to zero.

INTEL MNEMONIC(*): CLC

MNEMONIC	MODIFIER	DESCRIPTION
RSC		Reset Carry

INSTR	QUAL MOD	A		FLAG STATUS	DESCRIPTION OF OPERATION
RSC		F8	0->C	C0	Reset Carry

(*) Intel Corporation mnemonics copyright 1984

FORMAT: RTI

OPERATION:

```
PCL   <-  (SP)
PCH   <-  (SP + 1)
CSL   <-  (SP + 2)
CSH   <-  (SP + 3)
FL    <-  (SP + 4)
FH    <-  (SP + 5)
SP    <-  SP + 6
```

FLAGS: c,d,z,s,p

DESCRIPTION: Control is transferred back to the point of interruption by restoring the program counter (PC), code segment (CS), and flags (F) from the stack.

INTEL MNEMONIC(*): IRET

MNEMONIC	MODIFIER	DESCRIPTION
RTI		Return from Interrupt

INSTR	MOD			DESCRIPTION OF OPERATION
RTI		CF		Return from Interrupt

(*) Intel Corporation mnemonics copyright 1984

FORMAT: RTSx

OPERATION:

if called by JSR

```
PCL   <- (SP)
PCH   <- (SP + 1)
 SP   <- SP + 2
```

if called JSL

```
PCL   <- (SP)
PCH   <- (SP + 1)
CSL   <- (SP + 2)
CSH   <- (SP + 3)
 SP   <- SP + 4
```

FLAGS: none

DESCRIPTION: Transfers control from a subroutine back to the calling routing by pulling the return address from the stack into the appropriate registers.

INTEL MNEMONIC(*): RET

MNEMONIC	MODIFIER	DESCRIPTION
RTS		Return from Subroutine (JSR)
RTSL		Return from Subroutine (JSL)

			FLAG CONDITIONS (f)								DESCRIPTON OF OPERATION
			≥	≠	+	ODD	<	=	-	EVEN	ARITHMETIC
INSTR	MOD	UN	C0	Z0	S0	P0	C1	Z1	S1	P1	LOGICAL
RTS	UN	C3									Return from Subroutine
RTSL	UN	CB									Long Return from Subroutine

(*) Intel Corporation mnemonics copyright 1984

FORMAT: SBAx mod

OPERATION: A <- A - mod - c

FLAGS: c,d,z,s,p

DESCRIPTION: The modifier and carry flag are subtracted from A, and the result is placed in A.

INTEL MNEMONIC(*): SBB

MNEMONIC	MODIFIER	DESCRIPTION
SBA	R	Subtract R from A with Borrow
SBAN	(HL)	Subtract (HL) from A with Borrow
SBAI	d	Subtract d from A with Borrow

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
SBA	x	1AC0	1AC5	1AC1	1AC6	1AC2	1AC7	1AC3	1A07	1Cd	c,d,z,s,p	Subtract from Acc with Borrow

(*) Intel Corporation mnemonics copyright 1984

FORMAT: SEC

OPERATION: C1

FLAGS: C1

DESCRIPTION: The carry flag is set to one.

INTEL MNEMONIC(*): STC

MNEMONIC	MODIFIER	DESCRIPTION
SEC		Set Carry flag

INSTR	QUAL MOD	A		FLAG STATUS	DESCRIPTION OF OPERATION
SEC		F9	1->C	C1	Set Carry

(*) Intel Corporation mnemonics copyright 1984

FORMAT: STx leftmod, rightmod

OPERATION: leftmod <- rightmod

FLAGS: none

DESCRIPTION: The contents of the right modifier is placed into the the left modifier.

INTEL MNEMONIC(*): MOV

MNEMONIC	MODIFIER	DESCRIPTION
STAN	(HL)	Store A Indirect
STRN	(HL),R	Store R Indirect
STAD		Store A Direct
STPD	P	Store P Direct

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
STAN STRN STAD	(HL) (HL),x	 A2aa	882F	880F	8837	8817	883F	881F	8807			Store Acc Indirect Store Register Indirect Store Acc Direct

INSTR	QUAL MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
STPD	xx		890Eaa	8916aa	891Eaa	8926aa		Store Pair Direct

(*) Intel Corporation mnemonics copyright 1984

FORMAT: SUAx mod

OPERATION: A <- A - mod

FLAGS: c,d,z,s,p

DESCRIPTION: The modifier is subtracted from A, and the result is placed in A.

INTEL MNEMONIC(*): SUB

MNEMONIC	MODIFIER	DESCRIPTION
SUA	R	Subtract R from A
SUAN	(HL)	Subtract (HL) from A
SUAI	d	Subtract d from A

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
SUA	x	2AC0	2AC5	2AC1	2AC6	2AC2	2AC7	2AC3	2A07	2Cd	c,d,z,s,p	Subtract x from Acc

(*) Intel Corporation mnemonics copyright 1984

FORMAT: XCP leftmod, rightmod

OPERATION:
```
temp    <- leftmod
leftmod <- rightmod
rightmod <- temp
```

FLAGS: none

DESCRIPTION: The contents of the left modifier and the right modifier are switched.

INTEL MNEMONIC(*): XCHG

MNEMONIC	MODIFIER	DESCRIPTION
XCP	P1,P2	Exchange Pair, P1 <-> P2

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
XCP	BC,xx		87C9	87CB	87C8	87CC		Exchange BC <-> Pair
XCP	DE,xx		87D1	87D2	87D3	87D4		Exchange DE <-> Pair
XCP	HL,xx		87D9	87DA	87DB	87DC		Exchange HL <-> Pair
XCP	SP,xx		87E1	87E2	87E3	87E4		Exchange SP <-> Pair

(*) Intel Corporation mnemonics copyright 1984

FORMAT: XRAx mod

OPERATION:

```
A   <- A XOR mod
c   <- 0
```

FLAGS: C0,d,z,s,p

DESCRIPTION: A is exclusive ORed with the modifier. The result is placed in A. The carry flag is reset.

INTEL MNEMONIC(*): XOR

MNEMONIC	MODIFIER	DESCRIPTION
XRA	R	Exclusive OR R with A
XRAN	(HL)	Exclusive OR (HL) with A
XRAI	d	Exclusive OR d with A

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
XRA	x	32C0	32C5	32C1	32C6	32C2	32C7	32C3	3207	34d	C0,d,z,s,p	Exclusive OR x with Acc

(*) Intel Corporation mnemonics copyright 1984

DATA TRANSFER

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
LDA	x	8AC0	8AC5	8AC1	8AC6	8AC2	8AC7	8AC3	8A07	B0d		
LDR	B,x	8AE8	8AED	8AE9	8AEE	8AEA	8AEF	8AEB	8A2F	B5d		
LDR	C,x	8AC8	8ACD	8AC9	8ACE	8ACA	8ACF	8ACB	8A0F	B1d		
LDR	D,x	8AF0	8AF5	8AF1	8AF6	8AF2	8AF7	8AF3	8A37	B6d		Load with Register x
LDR	E,x	8AD0	8AD5	8AD1	8AD6	8AD2	8AD7	8AD3	8A17	B2d		
LDR	H,x	8AF8	8AFD	8AF9	8AFE	8AFA	8AFF	8AFB	8A3F	B7d		
LDR	L,x	8AD8	8ADD	8AD9	8ADE	8ADA	8ADF	8ADB	8A1F	B3d		
LDMNI	(HL),d									C607d		Load Memory Immediate
LDAD		A0aa										Load Acc Direct
STAN	(HL)								8807			Store Acc Indirect
STRN	(HL),x		882F	880F	8837	8817	883F	881F				Store Register Indirect
STAD		A2aa										Store Acc Direct

ARITHMETIC

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
ICA		FEC0									d,z,s,p	Increment Acc
ICR	x		FEC5	FEC1	FEC6	FEC2	FEC7	FEC3			d,z,s,p	Increment Register x
ICMN	(HL)								FE07		d,z,s,p	Increment Memory
DCA		FEC8									d,z,s,p	Decrement Acc
DCR	x		FECD	FEC9	FECE	FECA	FECF	FECB			d,z,s,p	Decrement Register x
DCMN	(HL)								FEOF		d,z,s,p	Decrement Memory
ADA	x	02C0	02C5	02C1	02C6	02C2	02C7	02C3	0207	04d	c,d,z,s,p	Add x to Acc
ACA	x	12C0	12C5	12C1	12C6	12C2	12C7	12C3	1207	14d	c,d,z,s,p	Add x to Acc with Carry
SUA	x	2AC0	2AC5	2AC1	2AC6	2AC2	2AC7	2AC3	2A07	2Cd	c,d,z,s,p	Subtract x from Acc
SBA	x	1AC0	1AC5	1AC1	1AC6	1AC2	1AC7	1AC3	1A07	1Cd	c,d,z,s,p	Subtract from Acc with Borrow

LOGICAL

INSTR	QUAL MOD	A	B	C	D	E	H	L	N (HL)	I	FLAG STATUS	DESCRIPTION OF OPERATION
ANA	x	22C0	22C5	22C1	22C6	22C2	22C7	22C3	2207	24d	C0,d,z,s,p	And x with Acc
XRA	x	32C0	32C5	32C1	32C6	32C2	32C7	32C3	3207	34d	C0,d,z,s,p	Exclusive OR x with Acc
ORA	x	0AC0	0AC5	0AC1	0AC6	0AC2	0AC7	0AC3	0A07	0Cd	C0,d,z,s,p	OR x with Acc
CPA	x	3AC0	3AC5	3AC1	3AC6	3AC2	3AC7	3AC3	3A07	3Cd	c,d,s,z,p	Compare x with Acc
CMA		F6D0										Complement Acc
CMR	x		F6D5	F6D1	F6D6	F6D2	F6D7	F6D3				Complement Register
CMMN									F617			Complement Memory Indirect

ACCUMULATOR AND CARRY CONTROL

INSTR	QUAL MOD	A		FLAG STATUS	DESCRIPTION OF OPERATION
CMC		F5	C->C	c	Complement Carry
RSC		F8	0->C	C0	Reset Carry
SEC		F9	1->C	C1	Set Carry
CLAC		32C0	0->A,0->C	C0,D0,Z1,S0,P1	Clear Acc and Carry
AJAD		27	A(adj)->A	c,d,z,s,p	Adjust A decimally
RLA		D0C0	A7->A0->A1...;A7->C	c	Rotate Acc Left
RRA		D0C8	A0->C;A0->A7->A6	c	Rotate Acc Right
RLAC		D0D0	A7->C-> A0->A1...	c	Rotate Acc Left with Carry
RRAC		D0D8	A0->C->A7->A6...	c	Rotate A Right with Carry

REGISTER PAIR INSTRUCTION

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
LDP	xx,CS		8CC9	8CCA	8CCB	8CCC		Load pair with CS
LDP	xx,DS		8CD9	8CDA	9CDB	8CDC		Load pair with DS
LDP	xx,SS		8CD1	8CD2	8CD3	8CD4		Load pair with SS
LDP	DS,xx		8ED9	8EDA	8EDB	8EDC		Load DS with pair
LDP	SS,xx		8ED1	9ED2	8ED3	8ED4		Load SS with pair
LDP	SP,xx		8BE1	8BE2	8BE3	8BE4		Load SP with pair
LDPI	xx		C7C1dd	C7C2dd	C7C3dd	C7C4dd		Load Pair Immediate
LDPD	xx		8B0Eaa	8B16aa	8B1Eaa	8B26aa		Load Pair Direct
XCP	BC,xx		87C9	87CB	87C8	87CC		Exchange BC <-> Pair
XCP	DE,xx		87D1	87D2	87D3	87D4		Exchange DE <-> Pair
XCP	HL,xx		87D9	87DA	87DB	87DC		Exchange HL <-> Pair
XCP	SP,xx		87E1	87E2	87E3	87E4		Exchange SP <-> Pair
PSP	xx	509C	51	52	53	54		Push Pair to Stack
PLP	xx	9D58	59	5A	5B	5C	(if AF:c,d,z,s,p)	Pull Pair from Stack
STPD	xx		890Eaa	8916aa	891Eaa	8926aa		Store Pair Direct

Figure B-3. 8088 Programming Aid

REGISTER PAIR ARITHMETIC

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
ICP	xx		41	42	43	44	d,z,s,p	Increment Pair xx
DCP	xx		49	4A	4B	4C	d,z,s,p	Decrement Pair xx
ADP	BC,xx		03C9	03CA	03CB	03CC	c,d,z,s,p	Add Pair xx to BC
ADP	DE,xx		03D1	03D2	03D3	03D4	c,d,x,s,p	Add Pair xx to DE
ADP	HL,xx		03D9	03DA	03DB	03DC	c,d,z,s,p	Add Pair xx to HL
ADP	SP,xx		03E1	03E2	03E3	03E4	c,d,z,s,p	Add Pair xx to SP

INPUT/OUTPUT INSTRUCTIONS

INSTR	MOD	AF	BC	DE	HL	SP	FLAG STATUS	DESCRIPTION OF OPERATION
IPA		E4p						Input to A from Port p
IPAN	(DE)	EC						Input to A Indirect
OPA		E6p						Output from A to Port p
OPAN	(DE)	EE						Output from A Indirect

MACHINE CONTROL INSTRUCTIONS

INSTR	MOD			DESCRIPTION OF OPERATION
NOP		90		No operation
HLT		F4		Halt
HLTT		9B		Halt until TEST* low

PROGRAM ADDRESS CONTROL INSTRUCTIONS

			FLAG CONDITIONS (f)								DESCRIPTON OF OPERATION
			≥	≠	+	ODD	<	=	-	EVEN	ARITHMETIC
INSTR	MOD	UN	C0	Z0	S0	P0	C1	Z1	S1	P1	LOGICAL
JPL	UN	EAaass									Jump Long
JPN	(BC)	FFE1									Jump Indirect, BC = 16-bit offset
JPN	(DE)	FFE2									Jump Indirect, DE = 16-bit offset
JPN	(HL)	FFE3									Jump Indirect, HL = 16-bit offset
JPR	UN	E9rr									Jump Relative, 16-bit Displacement
JPRS	f	EBr	73r	75r	79r	7Br	72r	74r	78r	7Ar	Jump Short on Condition f, 8-bit displacement
JSL	UN	9Aaass									Jump Subroutine Long
JSR	UN	E8rr									Jump Subroutine, 16-bit Displacement
RTS	UN	C3									Return from Subroutine
RTSL	UN	CB									Long Return from Subroutine

INTERRUPT INSTRUCTIONS

INSTR	MOD			
JI	d	CDd		Software Interrupt at Vector d
JI	3	CC		Software Interrupt at Vector 3
RTI		CF		Return from Interrupt

INTERRUPT CONTROL

INSTR	MOD			DESCRIPTION OF OPERATION
ENI		FB		Enable Interrupt Structure
DSI		FA		Disable Interrupt Structure

SEGMENT OVERRIDE INSTRUCTIONS

INSTR	MOD			DESCRIPTION OF OPERATION
ENS	CS	2E		Enable Code Segment Access
ENS	DS	3E		Enable Data Segment Access

Figure B-3. (Continued)

FORWARD RELATIVE OFFFSET

ADDRESS FOR 0 THRU 127 COUNTS

0	00	16	10	32	20	48	30	64	40	80	50	96	60	112	70
1	01	17	11	33	21	49	31	65	41	81	51	97	61	113	71
2	02	18	12	34	22	50	32	66	42	82	52	98	62	114	72
3	03	19	13	35	23	51	33	67	43	83	53	99	63	115	73
4	04	20	14	36	24	52	34	68	44	84	54	100	64	116	74
5	05	21	15	37	25	53	35	69	45	85	55	101	65	117	75
6	06	22	16	38	26	54	36	70	46	86	56	102	66	118	76
7	07	23	17	39	27	55	37	71	47	87	57	103	67	119	77
8	08	24	18	40	28	56	38	72	48	88	58	104	68	120	78
9	09	25	19	41	29	57	39	73	49	89	59	105	69	121	79
10	0A	26	1A	42	2A	58	3A	74	4A	90	5A	106	6A	122	7A
11	0B	27	1B	43	2B	59	3B	75	4B	91	5B	107	6B	123	7B
12	0C	28	1C	44	2C	60	3C	76	4C	92	5C	108	6C	124	7C
13	0D	29	1D	45	2D	61	3D	77	4D	93	5D	109	6D	125	7D
14	0E	30	1E	46	2E	62	3E	78	4E	94	5E	110	6E	126	7E
15	0F	31	1F	47	2F	63	3F	79	4F	95	5F	111	6F	127	7F

COUNT ZERO AT FIRST ADDRESS AFTER JUMP INSTRUCTION

BACKWARD RELATIVE OFFSET

ADDRESS FOR 1 THRU 128 COUNTS

1	FF	17	EF	33	DF	49	CF	65	BF	81	AF	97	9F	113	8F
2	FE	18	EE	34	DE	50	CE	66	BE	82	AE	98	9E	114	8E
3	FD	19	ED	35	DD	51	CD	67	BD	83	AD	99	9D	115	8D
4	FC	20	EC	36	DC	52	CC	68	BC	84	AC	100	9C	116	8C
5	FB	21	EB	37	DB	53	CB	69	BB	85	AB	101	9B	117	8B
6	FA	22	EA	38	DA	54	CA	70	BA	86	AA	102	9A	118	8A
7	F9	23	E9	39	D9	55	C9	71	B9	87	A9	103	99	119	89
8	F8	24	E8	40	D8	56	C8	72	B8	88	A8	104	98	120	88
9	F7	25	E7	41	D7	57	C7	73	B7	89	A7	105	97	121	87
10	F6	26	E6	42	D6	58	C6	74	B6	90	A6	106	96	122	86
11	F5	27	E5	43	D5	59	C5	75	B5	91	A5	107	95	123	85
12	F4	28	E4	44	D4	60	C4	76	B4	92	A4	108	94	124	84
13	F3	29	E3	45	D3	61	C3	77	B3	93	A3	109	93	125	83
14	F2	30	E2	46	D2	62	C2	78	B2	94	A2	110	92	126	82
15	F1	31	E1	47	D1	63	C1	79	B1	95	A1	111	91	127	81
16	F0	32	E0	48	D0	64	C0	80	B0	96	A0	112	90	128	80

COUNT ZERO AT FIRST ADDRESS AFTER JUMP INSTRUCTION

Figure B-3. (Continued)

8088 MNEMONIC	PRO-LOG MNEMONIC
ADC	ADC
ADD	AD
AND	AN
CALL	JS
XOR AL,AL	CLAC
CLC	RSC
CLI	DSI
CMC	CMC
CMP	CP
CS:,DS:	ENS
DAA	AJAD
DEC	DC
HLT	HLT
IN	IPA
INC	IC
INT	JI
IRET	RTI
JMP	JP
MOV	LD,ST
NOP	NOP
NOT	CM
OR	OR
OUT	OPA
POP	PLP
PUSH	PSP
RCL,ROL	RL
RCR,ROR	RR
RET	RTS
SBB	SB
STC	SEC
STI	ENI
SUB	SU
WAIT	HLTT
XCHG	XC
XOR	XR

Figure B-4. 8088-Structured Mnemonic Cross Reference

Cut Here

FOLD

FOLD

PRO-LOG® CORPORATION

Yes, I'm interested in Pro-Log.
Please send me more information about your:

- ☐ 7000 Series STD BUS Cards
- ☐ STD BUS Packaging Solutions
- ☐ STD BUS Subsystem
- ☐ STD BUS Systems
- ☐ Designer Software
- ☐ Design Courses and Seminars
- ☐ Programmer Product Line
- ☐ Other _____

- ☐ I am currently using Pro-Log equipment.
- ☐ Add my name to your mailing list.
- ☐ Please have a Sales Representative contact me.

Reader Comments: _____

Name _____ City _____
Company _____ State _____ Zip _____
Address _____ Telephone _____

PLACE
POSTAGE
HERE

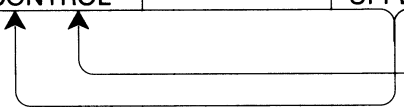
PRO-LOG®
CORPORATION
2411 GARDEN ROAD
MONTEREY, CA 93940

Attn: Literature Department

Notes

Notes

HEX				MSD	p=1	8	9	A	B	C	D	E	F
					p=0	0	1	2	3	4	5	6	7
	BITS				b8	p	p	p	p	p	p	p	p
					b7	0	0	0	0	1	1	1	1
					b6	0	0	1	1	0	0	1	1
LSD	b4	b3	b2	b5 / b1		0	1	0	1	0	1	0	1
0	0	0	0	0		NUL	DLE	SP	0	@	P	‘	p
1	0	0	0	1		SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0		STX	DC2	"	2	B	R	b	r
3	0	0	1	1		ETX	DC3	#	3	C	S	c	s
4	0	1	0	0		EOT	DC4	$	4	D	T	d	t
5	0	1	0	1		ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0		ACK	SYN	&	6	F	V	f	v
7	0	1	1	1		BEL	ETB	’	7	G	W	g	w
8	1	0	0	0		BS	CAN	(	8	H	X	h	x
9	1	0	0	1		HT	EM	)	9	I	Y	i	y
A	1	0	1	0		LF	SUB	*	:	J	Z	j	z
B	1	0	1	1		VT	ESC	+	;	K	[	k	{
C	1	1	0	0		FF	FS	,	<	L	\	l	|
D	1	1	0	1		CR	GS	—	=	M	]	m	}
E	1	1	1	0		SO	RS	.	>	N	^	n	~
F	1	1	1	1		SI	US	/	?	O	_	o	DEL
						CONTROL				UPPER CASE		LOWER CASE	



CONTROL CHARACTERS AND STANDARD KEYBOARD CONTROL SHIFT CHARACTERS

CTRL	@	NUL	Null	CTRL	K	VT	Vertical Tabulation	CTRL	V	SYN	Synchronous Idle
CTRL	A	SOH	Start of Heading	CTRL	L	FF	Form Feed	CTRL	W	ETB	End of Transmission Block
CTRL	B	STX	Start of Text	CTRL	M	CR	Carriage Return	CTRL	X	CAN	Cancel
CTRL	C	ETX	End of Text	CTRL	N	SO	Shift Out	CTRL	Y	EM	End of Medium
CTRL	D	EOT	End of Transmission	CTRL	O	SI	Shift In	CTRL	Z	SUB	Substitute
CTRL	E	ENQ	Enquiry	CTRL	P	DLE	Data Link Escape	CTRL	[	ESC	Escape
CTRL	F	ACK	Acknowledge	CTRL	Q	DC1	Device Control 1	CTRL	\	FS	File Separator
CTRL	G	BEL	Bell	CTRL	R	DC2	Device Control 2	CTRL	]	GS	Group Separator
CTRL	H	BS	Backspace	CTRL	S	DC3	Device Control 3	CTRL	^	RS	Record Separator
CTRL	I	HT	Horizontal Tabulation	CTRL	T	DC4	Device Control 4	CTRL	_	US	Unit Separator
CTRL	J	LF	Line Feed	CTRL	U	NAK	Negative Acknowledge			DEL	Delete

NOTE: The table above lists the keyboard character to be pressed with the Control key, the mnemonic for the function to be performed, and the description of the function. While the keyboard character is listed as uppercase, either uppercase or lowercase characters may be used.

ASCII Code Assignments

USER'S MANUAL

PRO-LOG®
CORPORATION

2411 Garden Road
Monterey, California 93940
Telephone: (408) 372-4593
TWX: 910-360-7082
Telex: 171879

111062-001A 1K 11/84